REAL TIME OPERATING SYSTEMS
WITH EMPHASIS ON THE MAN/MACHINE INTERFACE

A THESIS PRESENTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN CHEMICAL ENGINEERING

AT THE

UNIVERSITY OF CANTERBURY

CHRISTCHURCH, NEW ZEALAND

BY

M.G. RICHARDS

1982

ERRATA

Page	Line	
2-3	7	careful should read carefully
3-10	1	represent should read represents
4-3	6 up	insert it before is
6-10	10	transmission should read transmissions
7-5	1 up	insert no after is of
8-24	13 up	insert to after referred
8-29	5	insert the before terminal
	6	insert to be before significant
8-32	8	being should read was also
9-3	5	input should read inputs

~~with three fiche in back pocket~~

To my Father

"... it is a tale
Told by an idiot, full of sound and fury,
Signifying nothing."

Shakespeare

SUMMARY

The research examined distributed computer based process control systems.

A functional analysis was done to identify the subunits that make up such a system. They are:

i) a communication system

ii) a distributed database

iii) a presentation system for

  a) the process

  b) the process engineer

  c) the process operator

Each of these subunits is discussed in detail in chapters 3-7.

An economic analysis was also carried out to determine the optimum distribution of the control task. It was concluded that, on present costs, each controller should contain eight control loops. The controllers and man machine interfaces should be distributed along a multidrop communications line.

Based on these analyses, a system was implemented within the department. It used an M6800 microcomputer for the controller and the departmental PDP11 minicomputer for the man machine interfaces. The result proved to be effective, especially the process operator's interface. This used a colour graphics CRT with touch sensitive screen and ten key keyboard. This input system minimised training and, combined with the three level hierarchical displays, provided an efficient work station.

ACKNOWLEDGEMENTS

I wish, first of all, to thank Gordon Grey. Without him the touch pad and keyboard would have come to nought.

To my supervisor, Dr R.M. Allen, the staff and postgraduate students of the Chemical Engineering Department, and my family and friends, I offer my thanks for the numerous ideas and the encouragement.

This work was carried out with financial assistance from the Fletcher Holdings Postgraduate Scholarship.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Figures

1.1 What is Process Control?

The ultimate aim of a process control system is to maintain the process to which it is connected at its optimum operating state. This optimum state is determined by economic considerations combined with safety, raw material and plant constraints. Maintaining the process at its optimum, is effected by comparing the present state with the desired state and, by use of human operators, analog controllers or digital computers to remove any discrepancy. The transition path depends upon which of these methods is used. The human operators use experience based primarily on trial and error. Typically they are the least efficient. Analog controllers use variations of the PID (proportional, integral, derivative) algorithm. A digital computer can implement either of the above strategies and, in addition, methods based on optimal control theory and process models.

Put simply, a process can be considered as a black box (Fig. 1.1). The process has two classes of inputs; non-controllable, which are called disturbances, and controllable, some or all of which are used by the control system to counteract the deleterious effects of the disturbances. Some of the outputs from the process will be examined by the control system to determine the state of the process. These are compared with, or the computed resultant state is compared with, the optimum set by other controller inputs. Any difference causes the control system to alter the controllable process inputs so as to move the process closer to the optimum.

The control system has another function concomitant with maintaining the process at its optimum and that is information presentation. The process engineer and the process operator must be kept informed about the current state of the process. This may be achieved by displays on measurement apparatus such as thermometers or pressure gauges, by strip

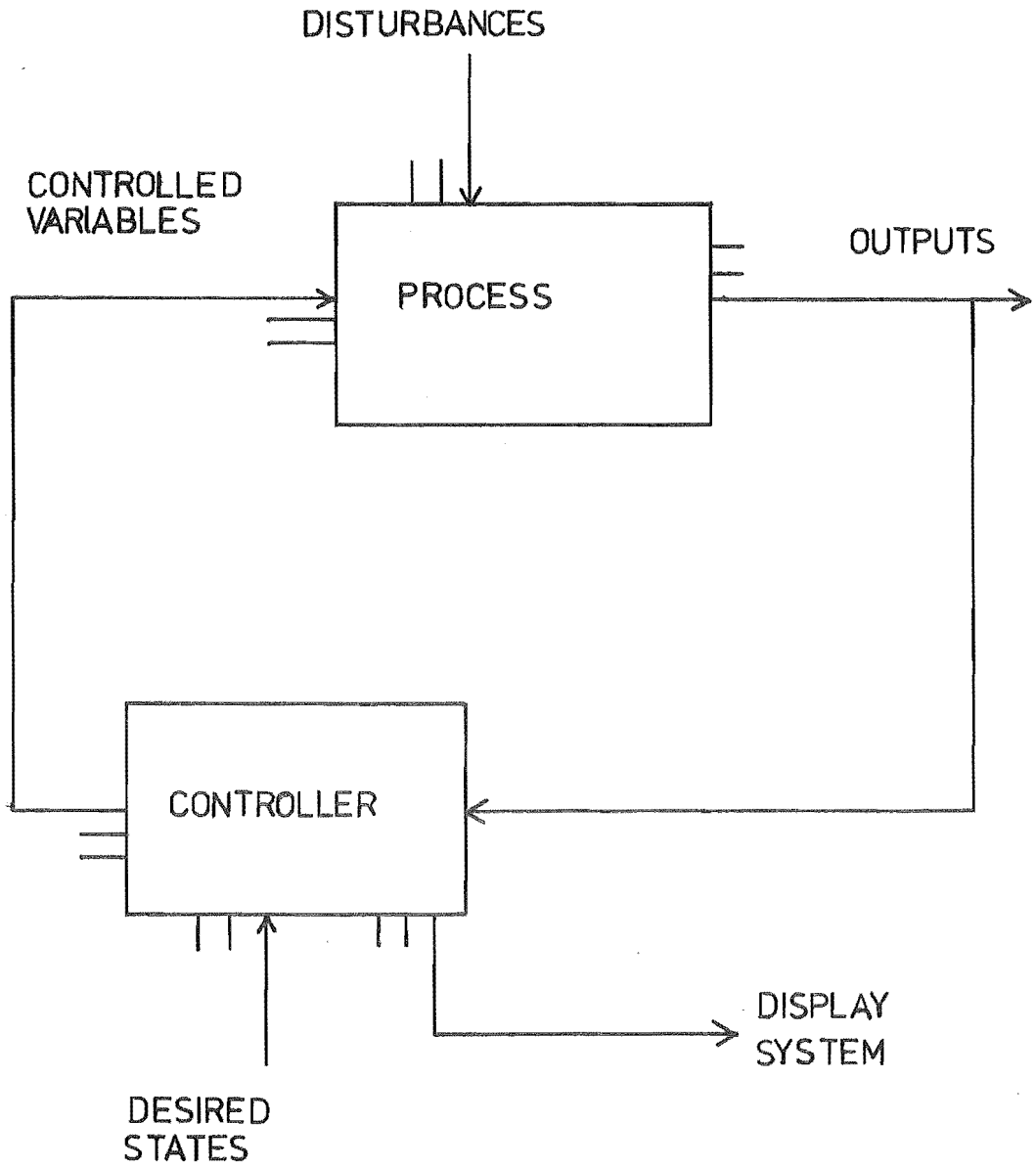


FIG 1.1 SIMPLIFIED CONTROL SYSTEM

chart recorders and panels on the front of controllers or, with modern technology, through VDU's (visual display units) and line printers connected to a computer.

1.2 Justification for Process Control

The equipment required to effect process control, the sensors, controllers, final control elements and such peripheral services as these use cost from 6% to 30% of the processing plant (1-5). As with any other expenditure this must be justified. Qualitative justification has been (1-5)

- a reduction in labour
- increased production
- improved and more uniform product quality
- reduced consumption of raw and process materials
- less plant maintenance under less severe operating conditions
- reduced utilities and energy consumption
- capital savings through
    - elimination of intermediate storage
    - reduced inventory
    - less excess capacity
- increased yields, minimum waste
- faster readjustment of the process to variations in raw materials or operating schedules
- increased plant and personnel safety

It is possible to directly quantify some of these factors, for example, reduced labour and plant requirements, savings on raw materials, utilities and energy and increased yields. It is also possible to place a value on those factors not so directly quantifiable. A plant catastrophe may cost $x. Without automatic control its probability may

be P_1 and with control this could be reduced to P_2. The saving is therefore $\$x(P_1 - P_2)$. This statistical argument can also be applied to increased product uniformity (1-5). These savings are then offset against the cost of the control system to decide if its use is economically justified.

1.3 Development of Process Control

Initially, process control consisted of sensors and their accompanying indicators or recorders geographically distributed around the plant. Large numbers of operators were required to monitor these and make the necessary adjustments to the control elements. Automatic controllers were developed because of the large variability in results between operators and in fact for the same operator depending on work load, time of day, mood etc. These controllers, P, PI or PID were still spread around the plant and during fault conditions an operator could still oversee only a small number of them.

To improve the efficiency of the operator, the next step was the design of centralised control rooms. The readings from sensors distributed around the process were fed to controllers housed in these control rooms and the resulting control action returned to the actuating element. The control rooms, typically, had the controllers spread along one wall, and while the operators now had their information close at hand, the volume presented resulted in important changes being missed. The use of human factors engineering, or ergonomics, to optimise the presentation of this information improved the situation. By logically grouping the information, using consistent displays and removing non-useful detail the workload on the operator was reduced. However, under high stress conditions information could not be assimilated quickly enough and performance degraded.

Human factors engineering, or the design of work situations to human capability constraints, was able to make further improvement when digital computers became cheap enough to allow their use in control rooms. This permitted the adoption of the "management by exception" principle to reduce operator overload. Parallel information presentation overwhelmed the operator by constantly displaying all available information about the process. Management by exception meant that the operator was only presented with information he needed at that particular time. This, combined with an alarm panel, significantly reduced the operator's load without reducing his efficiency. Figure 1.2 shows a typical operator display panel from Foxboro's IMPAC system (1-6). Figure 1.3 shows the same for a later model Honeywell VUPAC (1-7).

Because of their still high cost, the computer could not be justified solely in terms of man-process interface improvements. The machine was also time-shared around the control loops for either supervisory or direct digital control and data logging. This situation was far from satisfactory however, because computer failure meant loss of all loops. To reduce the catastrophic effect of this, the most critical loops had conventional analog controller backup. System reliability could be improved by using several computers, either reducing the area of responsibility of each or using the extra intelligence in a back-up mode. Here again, cost becomes a dominating factor and typically these types of computer control systems were and still are used only in highly critical areas, e.g. nuclear reactor control.

A solution through cheaper computer power became available with the advent of the microprocessor. Now it was possible to have many small computers, each responsible for as few as eight, or in rarer cases one, control loops. By distributing the microcomputers around the plant, near the item they were controlling, problems due to telemetry, or signal

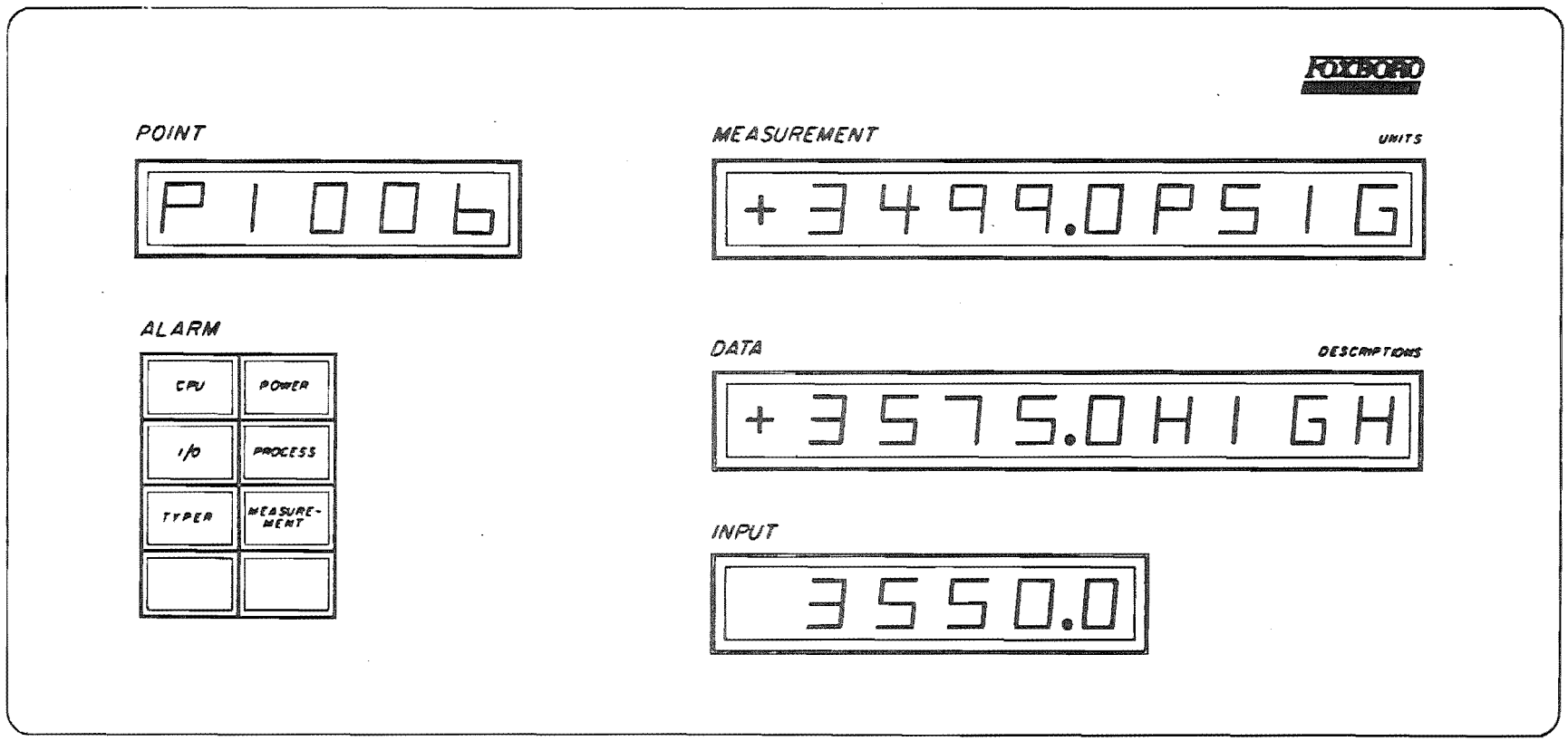
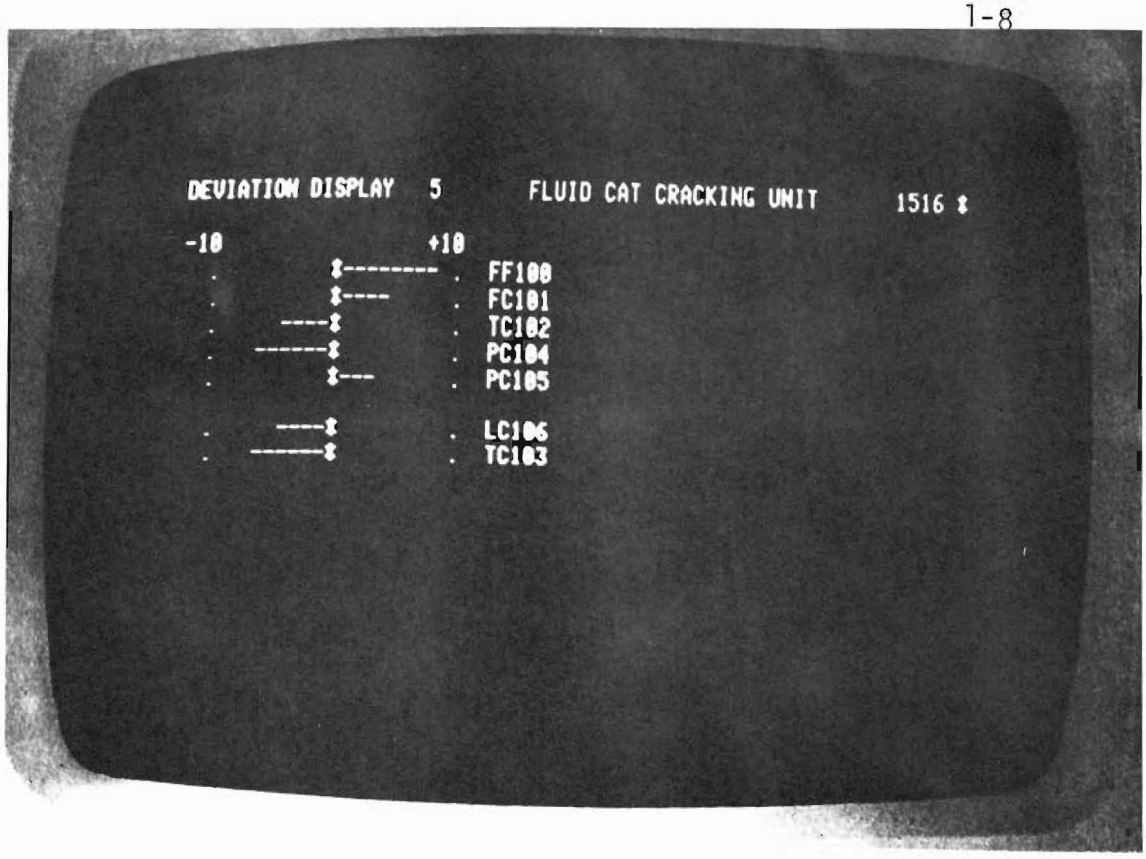


FIG 1.2 IMPAC OPERATOR INTERFACE



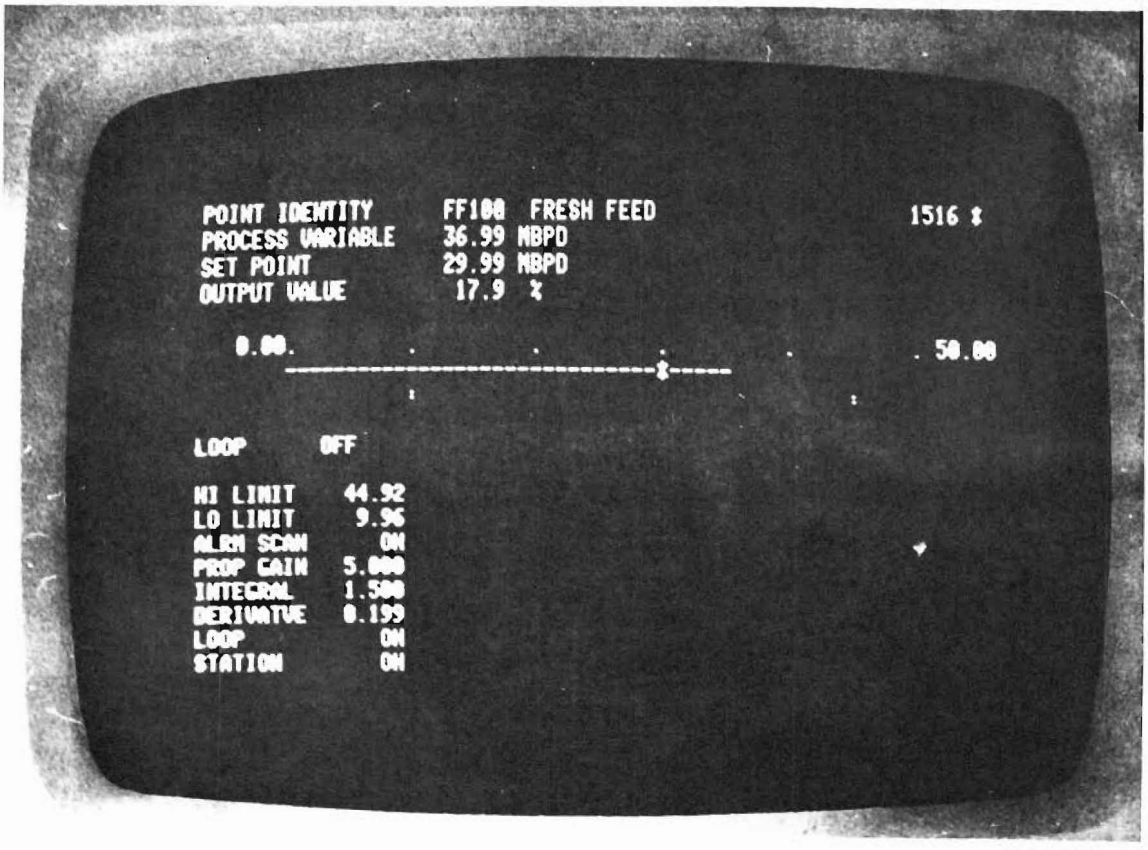


FIG 1.3 VUPAC OPERATOR INTERFACE DISPLAYS

transmission, such as noise were easily solved. Inter-computer communication was achieved over a twisted pair or coaxial cable, in one of three basic configurations (Fig. 1.4) i) a star

ii) a multidrop line

iii) a ring

The commercially available systems use variations or combinations of all three types.

1.4 Present Systems

Table 1.1 shows a list of the offerings commercially available at the time of writing. The table includes information about the controllers, the operator interfaces and, where appropriate, communications methods. These systems have been sorted into three categories based on their operational configurations (1-8).

Type 1 systems are the conventional centralized control room layouts. Sensor readings are brought into the control room and the output from the controller fed back to the actuator.

The multidrop line and the hierarchical star configurations make up Type 2 systems. The advantage here is that [illegible] level sensor signals have less distance to travel to the controller and thereby suffer less from the effects of noise. There is also greater use of the operator interface using the 'management by exception' principle (see section 1.3).

The last configuration, the ring, is the least used. There is only one control system, the System 1100 by Xomox, that uses it. Data transmission is uni-directional around the ring and hence reliability suffers.

Table 1.1 Current Control Systems

Type 1 Configurations

System	Company	Type	Controllers (all PID) Signal Conditioning	Alarming	Centralized Operator Interface	Hierarchical expandability
7000	Bailey Controls	P				
Bristol UCS3000	ACCO	E,μP			*?	
ac^2, dc^2	Fisher Controls	E			*?	
Spec 200	Foxboro	P,E,μP		*	*?	*
Centry	Leeds & Northrup	E	*	*		*
Synchro 350	Moore Products	E	*	*		
Diogenes	Rosemount	C	*	*	*G	*
Mod III	Sybron Taylor	E,C	*	*	*TOSM	*
Veritrak	Westinghouse	E	*	*	*?	*
NAF-Unic	Saab-Scania	μP				*
PCS 700	NV Philips	μP		*		*
Protronic	Hartman & Braun AG	E		*	*GLTAM	*

KEY

P = Pneumatic

E = Electronic

μP = Microprocessor

C = Computer

KEY

O = Overview

G = Group

L = Loop

T = Trend

A = Alarm

S = Status

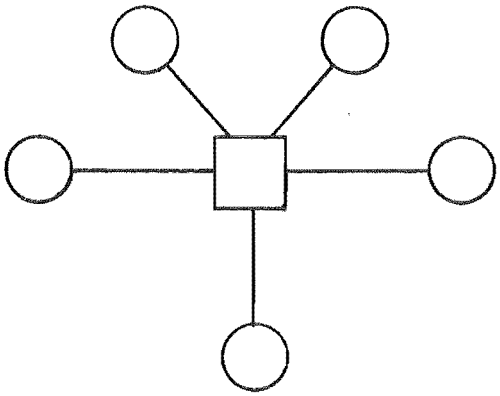
M = Mimic

All have local operator interfaces

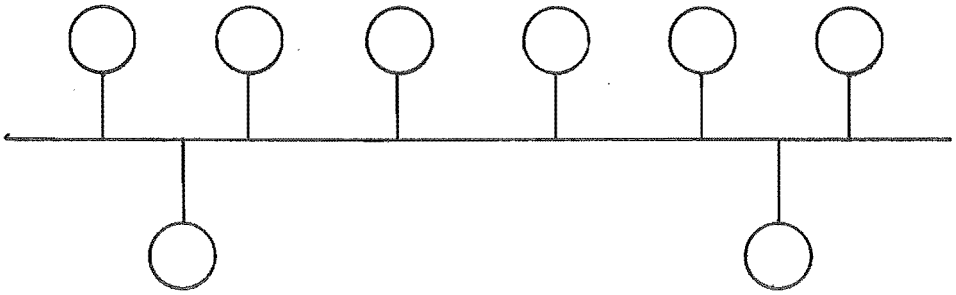
Operator Interface			Communication System		
Control	Local	Basis	Speed (baud)	Distance	Configurations
OGL	*	Current loop	?	?	Star
AGL		?			Multidrop
ROGLA	*	Coaxial	1.5M	5000 ft	Multidrop
TGO	*	Current loop			Star
?	*	?			?
OGL		Coaxial or twisted pair		600 ft	?
AOLMR		Twisted pair	500k		?
TG	*	?			?
AGLT	*	Coaxial		10 km	Multidrop
OGLTM	*	30 core cable			Star
?R		?	250k		?
OGLA	*	?		1000 ft	Multidrop & Star
?GR	*	?		15k ft	Multidrop
OGR	*	Coaxial	1M		Multidrop
OGLTR	*	RS232	300-19.2k		Star
MTLA		10 mA	1200-9600	3-.6 miles	Star
?G	*	Coaxial	1 M	3-4.5 km	Multidrop
?M		?	.5 M	4 km	Multidrop
OGLAM	*	Twisted pair	500 k	1500 m	Multidrop
?		Coaxial	500 k		?
-	-	Twisted pair	250 k	2 km	Multidrop
See Centum		RS232		10 km	Multidrop
		TTY			Ring

Key (Displays)

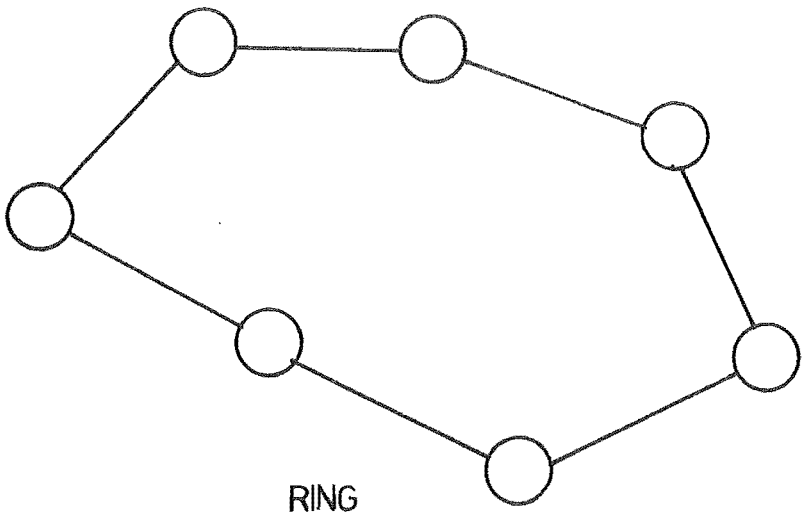
O = Overview
G = Group
L = Loop
T = Trend
A = Alarm
S = Status
M = Mimic



STAR



MULTIDROP LINE



RING

FIG1.4 DISTRIBUTED CONFIGURATIONS

1.5 Objectives of this Research

In view of the decreasing cost of computing power, the idea of distributing intelligence around the plant and providing a communications highway would become more prevalent in available control systems. This research aimed at outlining the requirements of these systems with special attention given to the problem of the man/machine interface.

In the following chapters the computer based distributed control system is first analysed to show the subsystems that are a part thereof, and then each subsystem is examined in greater detail. The final two chapters discuss a system that the author constructed based on the above information and the areas that could be improved with the availability of new or cheaper hardware.

REFERENCES

1-1 Handbook of Industrial Control Computers
T.J. Harrison, Wily-Interscience, 1972.

1-2 Man and Computer in Process Control
E. Edwards, F.P. Lees, I.Chem.E., 1972.

1-3 "Computerized Process Control in the Steel Industry"
Review 224, International Metals Reviews, 1977, pp.355.

1-4 "Control Equipment Developments Lead to a Future of Distributed Control"
E.J. Kompass, Control Engineering, December 1977, pp.35.

1-5 "The Justification of Expenditure for Process Control Equipment"
R.M. Allen, ACIS 8th National Conference, Auckland N.Z., 1977.

1-6 Fox 2 Process Computer Systems, Impac Software, System Product Specification SP-53000AX-A
The Foxboro Company, December 31, 1971, pp.IS35, IS37.

1-7 Vupak Systems Manual
Honeywell Inc., 1974.

1-8 "The Configurations of Process Control: 1979"
Control Engineering, March 1979, pp.43.

1-9 "International Manufacturers Offer Many Choices in Process Control Systems"
Kenneth Pluher, Control Engineering, January 1980, pp.54.

1-10 "Distributed System is Expandable from Eight Loop"
Henry H. Morris, Control Engineering, June 1980, pp.78.

1-11 "Introducing Four More New Integrated Distributed Control Systems"
Kenneth Pluher, Control Engineering, August 1980, pp.41-.

1-12 "Foreign Distributed Control Systems Keep Pace with Industry's Needs"
Henry H. Morris, Control Engineering, May 1981, pp.57.

CHAPTER 2

SYSTEM ANALYSIS

Figures

2.1 Functional Analysis

A functional analysis of any system involves answering the question, "What does the system do?" A distributed computer based process control system is designed to maintain a process at its optimum operating condition. "Maintain" implies the system must have a high reliability and "optimum" includes consideration of economic, safety and legal factors. Having broadly stated what the system does, the next question to be answered is, "How does it do this?".

To fully answer the second question, and thereby the first, we must break down the complex control system into its sub-units. These sub-units are then divided further until we are reduced to parts small enough to be explained in complete detail.

The type of distributed control system being studied has three separate interfaces with the world -

i) the process interface

ii) the engineer's interface

iii) the operator's interface

In systems with extensive report generating features, a fourth, the management interface, can also be defined.

The control strategy is implemented through the process interface. Sensors feed data to the control stations enabling the current state of the process to be determined. This state is then compared with a model stored in the microcomputer and any difference is used to calculate the action required to return to optimum. This action is then sent to the final control elements, valves, motors or heaters. In addition, checks on the approach of variables to safety and other constraints can be made and data for process logs can be generated.

The engineer, in designing the control strategy, enters information into the computer through his interface, enabling the computer to determine the state of the process and to calculate any action required.

A constant check on the state of both the process and the control system, and entry of data to fine tune both parts, is achieved through the operator's interface. Because of its critical function, the control system's success or failure as a whole depends on the careful considered design of this interface. SEE ERRATA

These three interfaces, or windows to the world, make up the presentation sub-unit. The second sub-unit that can be identified is the database or storage unit. The engineer's information, the control parameters, and the data collected from the process are stored in a database. The input from the operator's interface involves making adjustments to variables in this database. Its design affects the engineer's interface and thus the useability of the whole system and also its effectiveness as a control unit. The more efficient the database, the faster the system can respond to process changes. As a result, the database is viewed from two levels, the process interfaces with the physical storage of information and the engineer with logical storage of the same information.

In operation, data transfers of three types will be occurring -

i) storage-storage

ii) storage-presentation

iii) presentation-storage

These are made through the communication subsystem. Use of the communication channel for storage-storage transfers is necessary because of the distributed nature of the system. To fulful the reliability criterion, the control intelligence is not concentrated in one machine but distributed among several microcomputers. Consequently the database will also be spread over many machines making such storage-storage transfers necessary.

The communication subsystem is designed so as to minimize the cost of the overall scheme. Depending on the price of computing power, the resulting configuration will be a star, a ring or a multidrop line (see Fig. 1.4). If an economic analysis shows a small number of loops per processor to be optimal, the ring or multidrop line is preferred, while a large loop to processor ratio favours the star format. This is because of traffic density on the data highway. In the communications subsystem, consideration must be given to the physical unit, i.e. a twisted pair, a coaxial cable or an optic fibre, and the logical unit, i.e. the message format, error correction-detection scheme and the handshaking convention.

So we can now say that a distributed computer based process system is made up of the following subunits -

i) the communication system

ii) the distributed database

iii) the presentation system to the a) process
b) engineer
c) operator

It is also possible to consider the hardware subunits. These are the control station, the engineer's interface, the operator's interface, and the database and communication controller. Figure 2.1 gives a typical layout for the multidrop configuration.

2.2 Economic Analysis

Having completed a functional analysis, an economic analysis is necessary to answer the question, "How distributed should the system be?", and then decide which of the three possible configurations is more appropriate. To do this we need to find the most economic number of control loops for each microcomputer, or minimize a cost function with respect to the number of microcomputers for a fixed number of inputs.

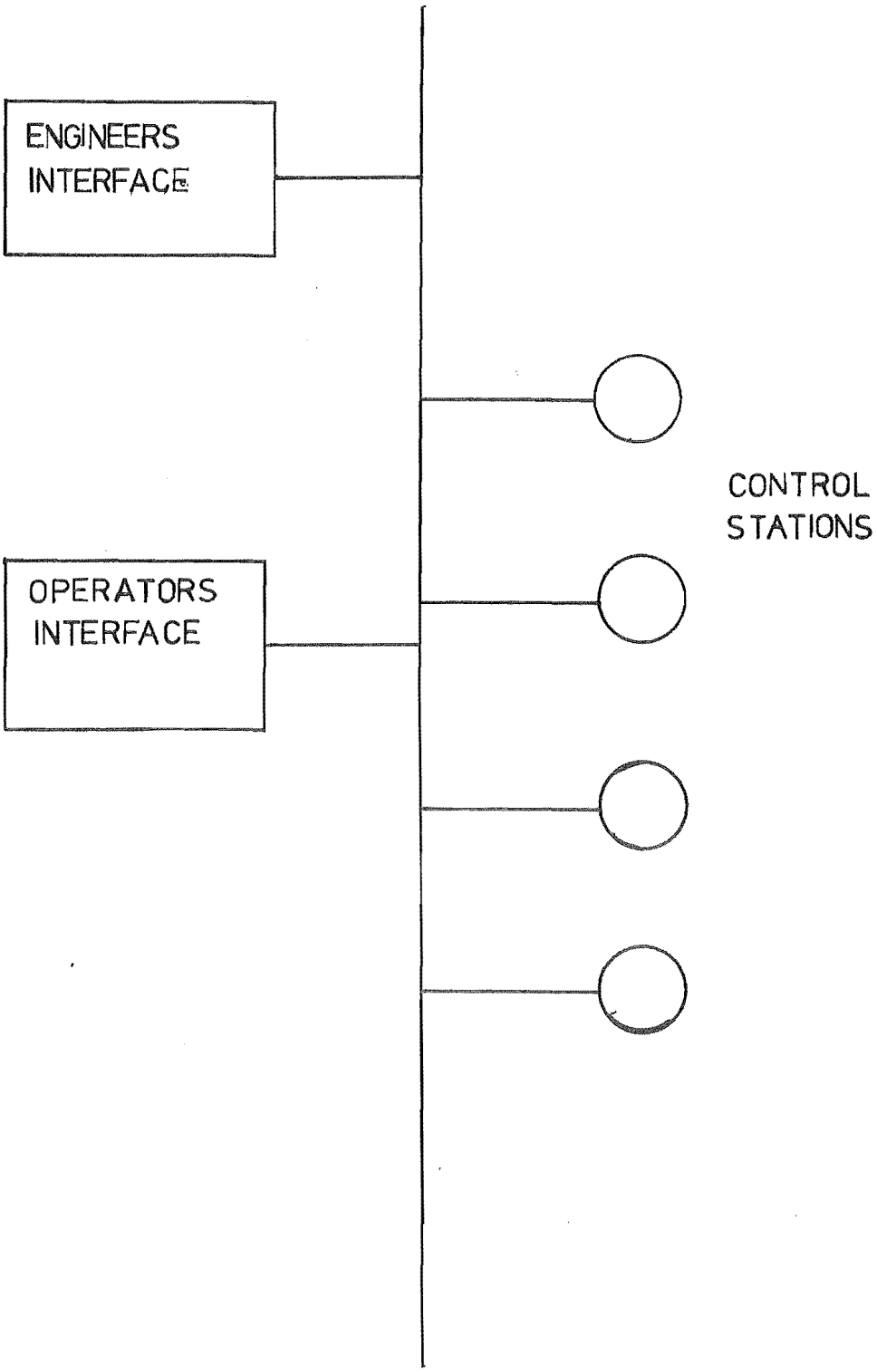


FIG2.1 MULTIDROP HARDWARE LAYOUT

The cost of a system includes -

i) design

ii) computer hardware and software

iii) wiring

iv) installation

v) failure

This analysis only considers the analog control system as for any particular situation the digital control system, best implemented through modern PLC's (programmable logic controllers), can be considered as a fixed cost. The logic control system will be required regardless of the number of loops that will be implemented in each microcomputer. If we assume a system with

N microcomputer based control stations

n inputs

and m outputs

where m is fixed at n/2, the same as that used for the Honeywell TDC2000 design, then for a fixed value of n, and therefore m, we vary N to minimise the cost. It is only necessary, therefore, to consider those factors that vary with N as all others will effectively be only a fixed additional cost. From our original list, then, we may ignore design and installation costs which leaves four sections -

i) hardware

ii) software

iii) wiring

iv) failure.

2.2.1 Hardware costs

Hardware cost = CPU + memory + input/output + power supply + box, cabling, etc.

Let H = CPU + power supply + box, etc.

Now memory comprises program and data memory, i.e. ROM + RAM. The program memory will be of fixed size, but the data memory will have a fixed requirement plus extra for control loop data storage. If we allow 5k bytes of memory for the storage of program and set data, and allow .5k bytes for each control loop, the memory requirement becomes

$$(5 + .5n/N)k \text{ bytes}$$

The input/output subsystem includes the communication system, process input and process output. If we now let

$$H' = H + \text{communication system}$$

then the hardware cost per microcomputer

$$= H'' + (10 + R)*C_m + [\frac{R}{8}]*C_{DA} + \frac{R}{2} * C_D$$

where H'' = the cost of H'

C_m = the cost of .5k bytes of memory

C_{DA} = the cost of an eight channel data acquisition module

C_D = the cost of a D/A converter

R = the number of inputs per microcomputer = n/N

and $[\ ]$ = the ceiling function.

Therefore, the system hardware cost

$$C_H = (H'' + 10*C_m)*N + [\frac{R}{8}] * C_{DA} * N + R * C_m * N + \frac{R}{2} * C_D * N$$

$$= (H'' + 10*C_m)*N + [\frac{n}{8}] * C_{DA} + n * C_m + \frac{n}{2} * C_D$$

c.f. $y = aN + b$ where b is only piecewise constant. (See Fig. 2.2).

2.2.2 Software costs

In every microcomputer there will be a set of standard routines that are used to effect the control. These will be the same for each control station and so will not affect the cost function. The software costs that should be considered are those of a load sharing routine, to even out the number of inputs scanned each interval, if the number of inputs per microcomputer becomes too large, and if only one large machine is used, a comprehensive multitasking executive. If we assume the basic routines cost S, the load sharing routine for greater than 16 inputs per microcomputer is S_L, and the multitasking executive for N = 1 is S_E, the result is shown in Figure 2.3.

2.2.3 Wiring costs

Let us assume that the microcomputer based control stations lie on a circle of radius D. The inputs to each station will be connected in a star configuration and can be approximated to lie on a circle of radius r about that station. Then the length of cable needed to join the control stations is ND for the star

$2\pi D(1 - \frac{1}{N})$ for the multidrop line

and $2\pi D$ for the ring,

and the length of wire needed to join the inputs to their respective station is $\frac{nr}{N}$. As the number of stations gets larger each input will be closer so $r \alpha \frac{1}{N}$, say $r = \frac{k}{N}$. Therefore the wire for all input is

$$\frac{nr}{N} * N = \frac{kn}{N}$$

If the wire for joining the control stations costs x per unit length and for the inputs y per unit length, the total wiring cost is

$$C_w = ND_x + \frac{kny}{N} \qquad \text{star}$$

$$= 2\pi D(1 - \frac{1}{N})x + \frac{kny}{N} \qquad \text{multidrop line}$$

$$= 2\pi D_x + \frac{kny}{N} \qquad \text{ring}$$

c.f. $y = aN + \frac{b}{N}$ star

$= a + \frac{b}{N}$ multidrop line or ring

(see Fig. 2.4).

2.2.4 Failure costs

Let the cost of failure of a single control loop be c per hour. If the number of loops for control station is $\frac{n}{N}$, then the cost of microcomputer failure is $\frac{nc}{N}$ per hour. Assuming a negative exponential failure distribution (2-1) and defining

F = mean time to failure

R = mean time to repair

T = operating life in hours (>> F),

the downtime of a microcomputer is

$$\frac{RT}{R + F} \text{ hrs.}$$

Therefore the cost of failure is

$$\frac{nc}{N} * \frac{RT}{R + F} .$$

Now ncT is the cost of failure of all loops for the operating life of the control system. This is equivalent to having no control at all, and under these circumstances the plant will cease to be profitable. Therefore ncT equals the profit of the plant, say P. The cost of failure then becomes

$$\frac{P}{N} * \frac{R}{R + F}$$

and for all control stations

$$= P * \frac{R}{R + F}$$

which is independent of N.

However, as the number of control stations decreases, the number of components used to construct each station will increase and this will affect F, the mean time to failure. So let F = a - bx, where

a and b are constant and x is the number of additional components required. Then the cost of failure

$$C_F = P * \frac{R}{R + (a - bx)}$$

c.f. $y = a N^b$

(see Fig. 2.5).

2.2.5 Conclusions

To determine the optimum number of inputs per control station (or the optimum number of stations for the system), the total system cost must be minimised. As stated in section 2.2, this total cost, considering only the parts that vary with the number of control stations N, is

Total cost = hardware + software + wiring + failure.

The equations were programmed on a computer using the following data:-

i) Hardware costs based on current N.Z. retail prices

$H'' = \$1,000$

$C_m = \$50$

$C_{DA} = \$150$

$C_D = \$10$

ii) Software costs

The loading sharing routine is approximately 500 statements long (as designed by the author) and the executive (based on the PDP11 RT-11 system) is 2000 statements. At a cost of $6 per statement (2-2), the costs are

$S_E = \$12,000$

$S_L = \$3,000$

iii) Wiring costs

A typical process plant of 200 loops will cover an area

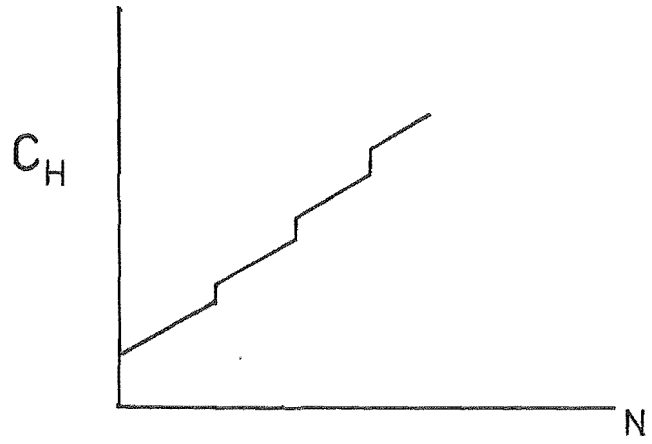


FIG 2.2 HARDWARE COST

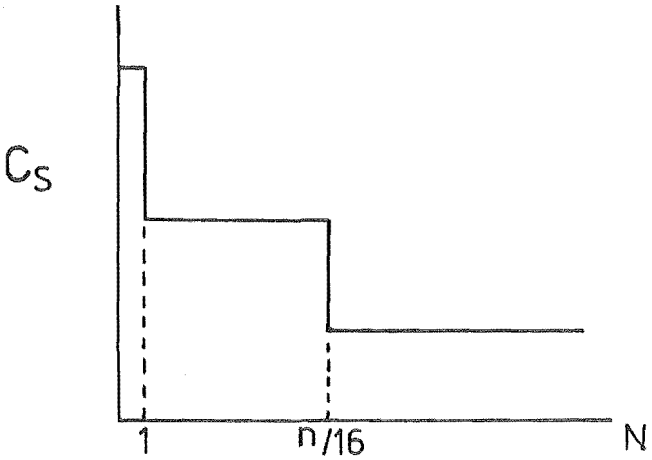


FIG 2.3 SOFTWARE COST

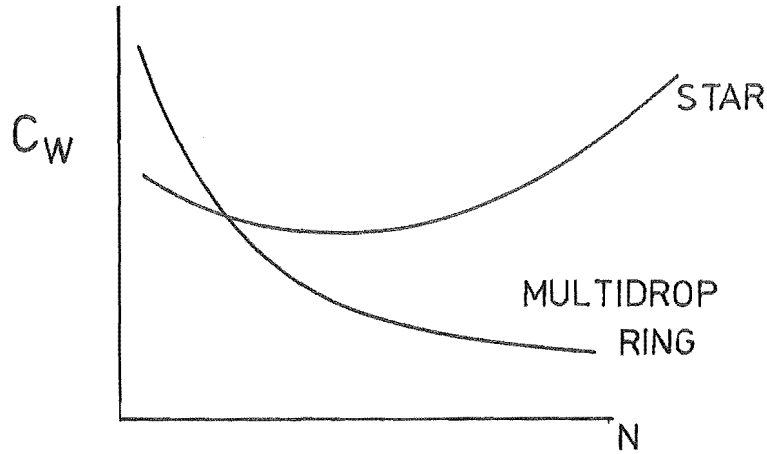


FIG 2.4 WIRING COST

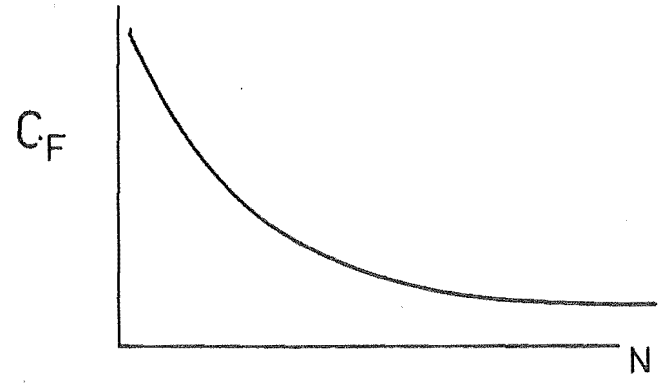


FIG 2.5 FAILURE COST

of diameter of the order of one kilometre, as D = 1000. Coaxial cable, for inter-station connection is $1 per metre on current costs and for input connection, cheaper wire of 30¢ per metre is adequate. For the sake of simplicity the constant k is set equal to D.

iv) Failure costs

The profit of a plant can be considered as the return on an investment equal to the capital cost of that plant, so

$$P = A(1 + \frac{r}{100})^n$$

where n = plant life in years, 15

r = effective rate of return, 15%

A = capital cost, $\$10^8$

Therefore P = $\$8.14 \times 10^8$.

The mean time to repair a modern computer board is of the order of one hour so R = 1. The mean time to failure of the same is 1000 hours, and considering the extra complexity each additional chip will reduce this by 5 hours. Strictly speaking this time is not constant and will exponentially decrease but for the range considered in the program it was assumed linear.

The results are shown in Table 2.1.

The first important conclusion is that the star configuration is never the most economic. The difference between the multidrop line and ring format is not significant and these have been included in the one column. The ring suffers from reliability (see Chapter 6) without the addition of intelligent hardware and so is less favoured than the multidrop line.

Table 2.1: Economic Analysis Results

Inputs	Optimum Inputs/μC		Total Cost	
	Star	Multidrop Line	Star	Multidrop Line
40	4	3	9.19×10^5	9.14×10^5
80	5	4	9.44×10^5	9.34×10^5
120	6	5	9.65×10^5	9.50×10^5
160	7	6	9.83×10^5	9.65×10^5
200	8	7	9.98×10^5	9.78×10^5
240	8	8	1.01×10^6	9.90×10^5
280	8	8	1.03×10^6	1.00×10^5
320	8	8	1.04×10^6	1.02×10^5
360	8	8	1.06×10^6	1.02×10^5
400	8	8	1.08×10^6	1.04×10^5

The range of inputs considered was a minimum of 40 to a maximum of 400 per system. The optimum always lay between 2 and 16 inputs per microcomputer and so neither extra software routines affected the cost. The most significant cost was the failure cost, contributing as much as 90% of the total cost in some cases. The wiring and hardware costs were of a similar order and for the largest system were only 40% of the overall total. Ignoring the failure cost completely moved the optimum to more inputs per station, stopping at 16 because of the additional software overhead of the load sharing routine.

REFERENCES

2-1 System Reliability Engineering

G.H. Sadler, Prentice Hall, 1963.

2-2 'Focus on Software'

M. Schindler, Electronic Design 6, 1979, pp103.

CHAPTER 3

PROCESS INTERFACE

Figures

3.1 Function

The process interface has two identifiable functions. The first is to effect the control strategy as defined by the engineer. The second is to collect and process data for use by the process operator in his role as supervisor, or by the engineer, in reports or higher level control calculations. The interface must therefore be able to capture this data, convert it to a form suitable for use by the microcomputer, process the result using the available software routines or computer language, and finally output the desired control action and convert it to a signal capable of driving the control actuating device (Fig. 3.1). The hardware components, including the microcomputer, are discussed in section 3.2 and the software requirements in section 3.3.

3.2 Hardware

What must the hardware section of the process interface accomplish? It must measure the physical variables of interest, convert these to an electrical and/or digital analogy which can be used by a microcomputer based controller. On the output side it must do the reverse. The numerical value from the μC is changed to a signal suitable for actuating the final control element.

The input can be considered as having four separate stages (Fig. 3.1):

i) the transducer

ii) pre-multiplexor conditioning

iii) the multiplexor

iv) sample and hold/analog digital converter

Table 3.1 lists the more common process variables and means of measurement. It should be noted that not all are real time methods and some do not give a direct conversion from the physical variable to an electrical analogy.

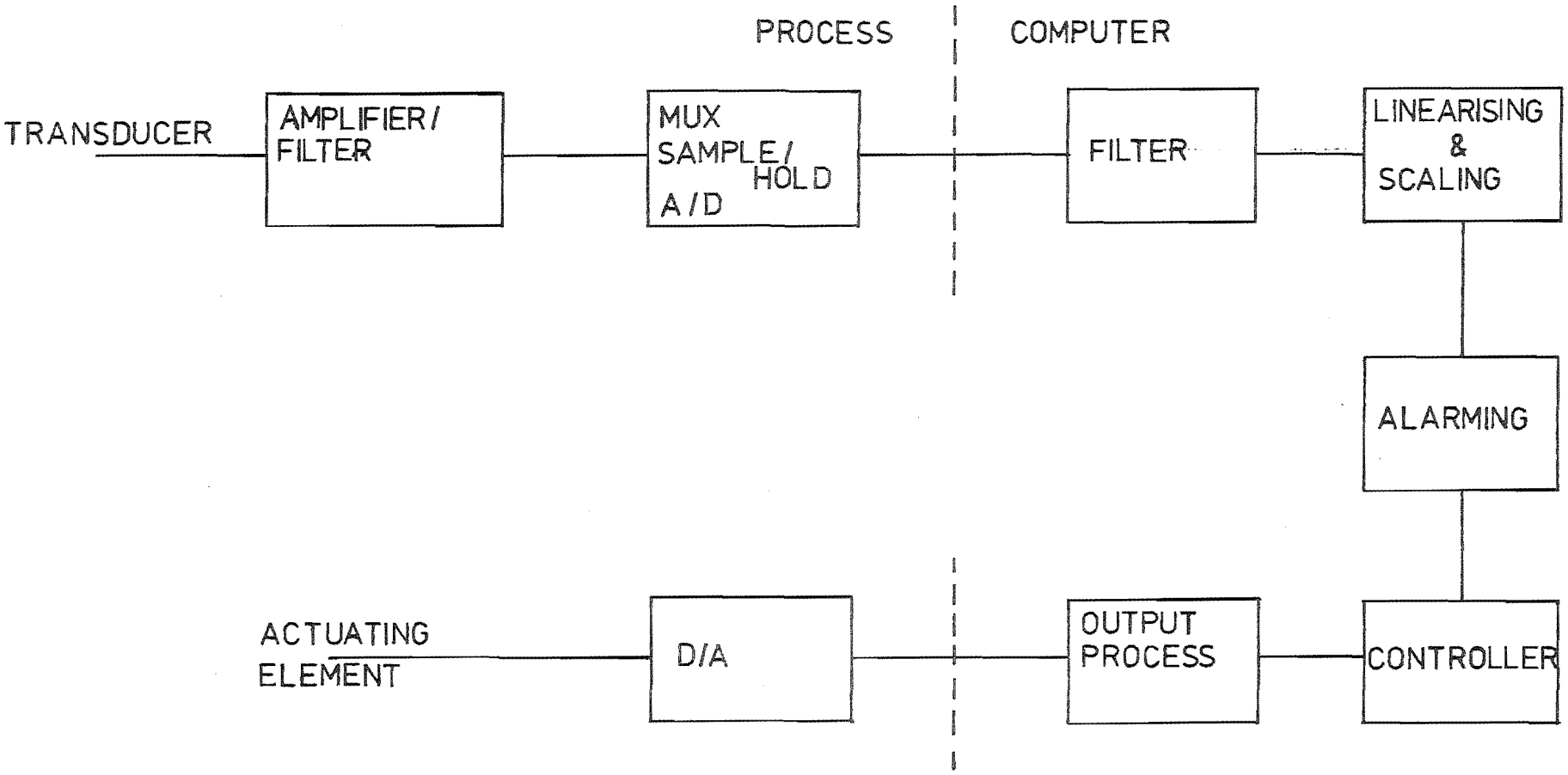


FIG 3.1 PROCESS INTERFACE

The particular choice of transducer depends on three factors:

i) economy

ii) safety

iii) accuracy, relative and absolute

Economic constraints are important but are not the overriding criterion. Safety aspects may preclude the use of certain transducers. For example, it may be necessary to convert a level measurement to a pneumatic signal first because the presence of an electrical signal may constitute an explosion hazard. Transducer accuracy is affected by its two basic parts. To measure a variable, the transducer compares the incoming signal with some previously determined standard. The quality of this reference affects the absolute accuracy of the measurement and the comparison affects the relative accuracy. So it is possible for the device to be consistent (precise or relatively accurate) but to give a poor indication of the state of the physical variable (inaccurate or poor absolute accuracy). The resolution of the comparing mechanism will also affect accuracy. Low resolution means the available scale is divided into very few stages giving a coarse measurement.

Having eventually converted the process variable into its electrical analog, the signal must be prepared for computer use. If the computer is any distance from the measurement point, the signal must be transferred. This transfer gives rise to a large number of errors. To reduce the effect of noise on the signal it should be amplified as soon as possible. This amplification must take into account such problems as common mode error sources, normal mode error sources and the major causes of these. Common mode errors are, typically, voltages that are in parallel with the desired signal. These can arise due to ground loops or induction between the signal carrying wires and nearby power sources or users. Normal mode voltages are produced in series with the signal and are less easy to deal with. They

Table 3.1: Process Variables and Measurement Systems

- Temperature
  - Thermocouples
  - Resistance thermometers
  - Filled system thermometers
  - Liquid-in-glass thermometers
  - Bi-metallic strips
  - Optical pyrometers
- Pressure
  - Liquid column
  - Elastic element methods
    - Bourdon tube
    - Bellows
    - Diaphragms
  - Electrical methods
    - Strain gauges
    - Piezoresistance
    - Piezoelectricity
- Flow
  - Variable head meters
  - Variable area meters
  - Positive displacement meters
  - Turbine meters
  - Mass flowmeters
  - Sound reflection
  - Light reflection
  - Weirs
  - Flumes
- Level
  - Visual gauges
  - Float actuated
  - Displacer
  - Head
  - Electrical methods
    - Resistance
    - Capacitance
  - Sound reflection
  - Nuclear radiation
  - Light reflection
  - Thermal conductivity
- Physical Properties
  - Density and Specific Gravity
    - Liquid column
    - Displacement
    - Direct mass
    - Radiation absorption
    - Viscous drag
  - Viscosity
    - Viscous drag
  - Refractive index
    - Transmission
    - Reflectance
  - Thermal conductivity
  - Boiling point
  - Flash point

Table 3.1 contd.

Chemical Composition

- Chromatographic analyzers
- Infrared analyzers
- UV and visible analyzers
- Turbidimetry
- Paramagnetism

Moisture

- Dew point
- Electrolysis
- Capacitance
- Heat of sorption
- Piezoelectricity
- Psychrometry
- Conductance
- Karl Fischer method

Electroanalyticity

- Conductance
- Capacitance
- Potential
- pH
- Specific ion probes
- Colorimetric analysis

may arise due to resistive drops in signal wires, thermocouple effects when dissimilar wires are joined and, again, ground loops. Common mode problems can be solved, or eased, by using instrumentation or isolation amplifiers, (3-3), or early digitizing, but correction of normal mode errors can only be effectively done by careful design to remove their sources. The amplified signal will still have some noise and this can be reduced by filtering. In addition, the use of a filter can prevent aliasing or errors due to frequency folding. Aliasing is caused by high frequency components of the signal. A slow sampling rate can cause the computer to react to harmonies of these higher frequencies, instead of the signal of interest. Frequency folding is an effect caused by the reflection of the higher frequencies about a point equal to half the sampling frequency. If the highest frequency in the signal is f_h and the sampling rate if f_s, then if $(f_s - f_h)$ is less than $f_s/2$, there will be a resultant frequency folding (3-11). Typically, a low pass filter with a cutoff frequency as low as possible, commensurate with the bandwidth of the signal of interest, is used, but specialized filters such as notch filters, synchronous filters or derivative controlled filters may be required in some situations. For example, where specific noise spikes occur (notch or synchronous filters), or where there may be a speed of response requirement (derivative controlled filters) (3-3).

Primarily for economic reasons, the now amplified and filtered signal is fed to a multiplexor (MUX). The use of a MUX means that only one analog to digital converter (A/D) is needed for many incoming measurements. The MUX is a set of switches that can be selectively turned on to connect only one measurement line to the A/D. The switches can be mechanical, as in relays, or semiconductor, as in FET switches. The problems in using a MUX are:

i) leakage, due to off resistance being less than infinite;

ii) crosstalk, the off channel affects the selected channel giving common mode problems;

iii) switching time, affecting the possible sampling rate.

The last step before the computer is A/D conversion. The choice of converter is critical to the whole measurement. Its resolution, accuracy and speed affect the final result.

There are two basic types used in process control, an integrating A/D and a successive approximation type. The integrating converter integrates the incoming signal for a set time interval and then measures the time required to ramp the result back to zero at a fixed rate. Typically they are only used in low speed applications. The successive approximation variety compares the signal to an internal voltage created by successively turning on bits in the resulting word and leaving them on or switching them off after comparing the two values. They are much faster and have the capacity for high resolution. However the A/D is typically not fast enough to convert the signal without it having changed appreciably during the conversion. For example, a 12 bit A/D that had a conversion time of 24 μsecs could only handle a signal of 1.6 Hz. To overcome this problem a sample and hold unit is used before the A/D to sample the signal and hold the result for the A/D to convert. If it is necessary to sample a large number of channels simultaneously, the sample and hold unit may be placed before the MUX. Accuracy in an A/D is dependent on its linearity, as zero and full scale offset voltages can be trimmed with a resistance. A typical specification will quote a total non-linearity of plus or minus a fraction of the least significant bit and at any stage the output should be in error by no more than this. The process variable has now been measured and converted to a computer useable form. Other input conversion systems may be used however. For example, a

voltage to frequency converter may be preferable with the result being measured by a pulse counter. This can reduce, in some situations, the effects of some of the errors we have discussed.

After the software has processed the input, its digital result must be delivered back to the actuating device. The necessary conversion is done by the output hardware. The first step is to convert the digital result into form acceptable to the actuator of the final control element. This may mean conversion to an electrical value via a digital to analog (D/A) converter, or to a form of modulated pulse output.

The two main types of digital to analog converters are the weighted current source and the R-2R ladder network. The weighted current source D/A has a series of transistors whose collector currents are set by resistors with values of R, 2R, 4R etc. The transistors are switched on or off according to the appropriate bit in the output word. This method is fast and simple but the large range of resistance values required for high resolution results in significant temperature effects. The R-2R network has a series of resistances that are either in series with the output of values R or shunt across the output with values 2R. The shunts are connected to earth or a voltage source depending on the bit value of the output and the division of current as it flows down the ladder results in the required output. This method requires twice as many resistors as the weighted current source but all are of value R or 2R and so are better matched. Other advantages are that the output amplifier always sees a constant input resistance and the resistance values can be kept low to ensure low power consumption and high speed.

The pulse output systems can be modulated in one of four ways:

i) amplitude

ii) frequency

iii) duration

iv) phase lag.

For computer output that represent the absolute value of the control element, amplitude modulation tends to be the most common as it is the least complicated to implement. For change in position output, any of the four methods can be used.

This conditioned output is then transmitted to the actuating element of the controller hardware. Table 3.2 lists common actuating methods and final control elements (3-2). As with the input, the use of pneumatics or hydraulics is typically for safety reasons although, with some output situations, they may also effect a large amount of power amplification.

The remaining hardware unit in the process interface is the control station, or microcomputer, itself. Its primary design consideration is reliability. Less important criteria are performance, cost and modularity. The three functions it must perform are control, communication and the provision of a local man-machine interface. To some extent these are independent and it is therefore possible to allocate each to a separate CPU, fulfilling all but the cost restriction. However it is cheaper to use only a dual CPU system with little or no loss in performance, modularity and reliability. Figure 3.2 shows the result.

Reliability, and reconfiguration should a fault be detected, are both accounted for. The control interface CPU is capable of stand-alone operation and the unit as such could be used as a multiloop controller. Therefore communication system failure should not affect control . Should the control CPU develop a fault, the communication CPU, through the duplicated process inputs, can send the measurements to a standby CPU with the results being sent back for output. This reconfiguration should take place in less than ten sample intervals and if the communication channel is fast enough in less than a second. The layout is therefore dynamically reconfigurable giving a very high reliability for small cost (redundancy is reduced from complete hardware duplication to only a small number of

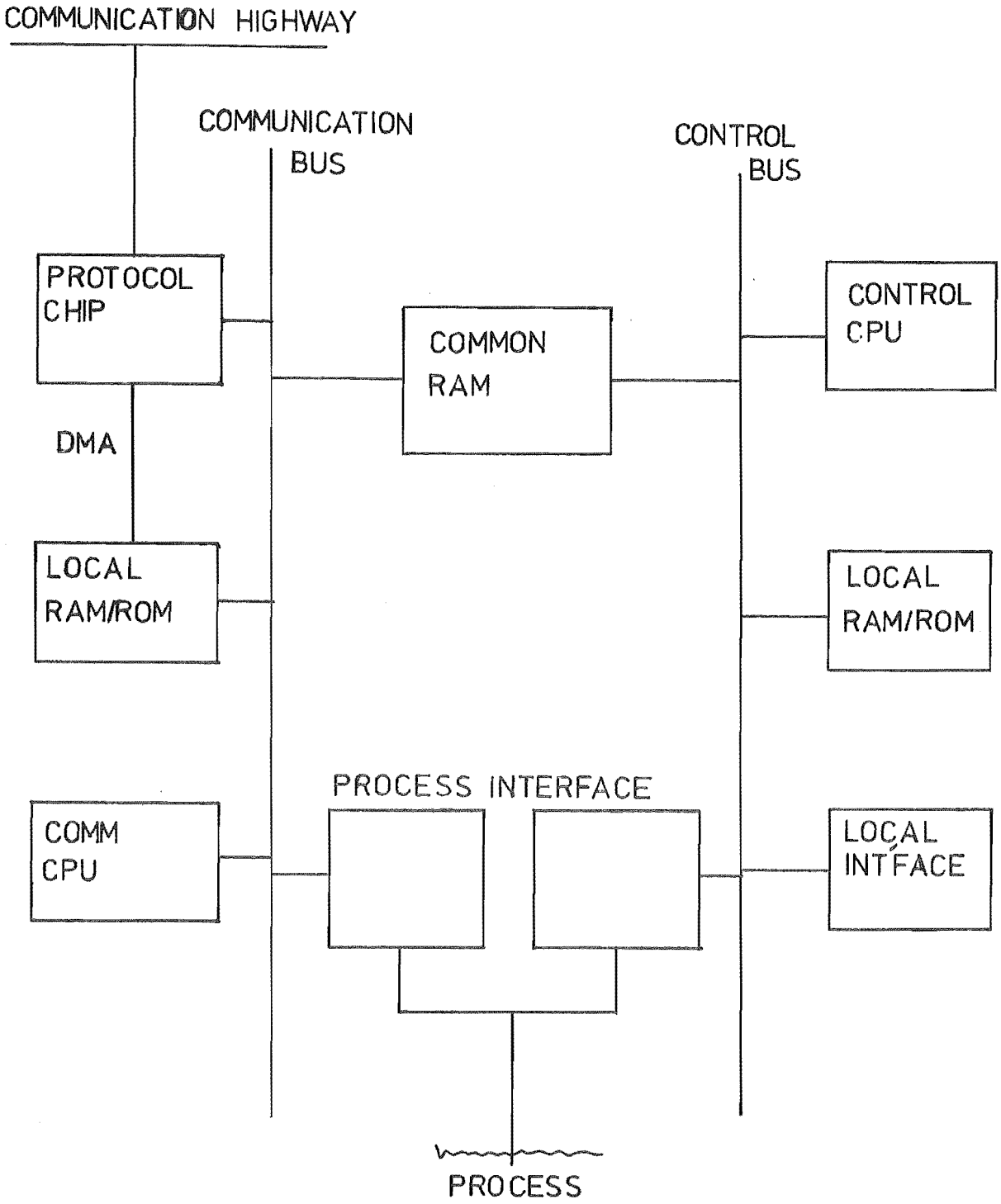


FIG3.2 CONTROL STATION CONFIGURATION

standby controllers).

The other criteria are also satisfied with this design. Cost is small, as explained above, the only extra requirement being fast access memory so the CPU's can interleave their read/write cycles. Performance is enhanced by using dedicated intelligence for each function and the modularity constraint is complied with as explained above.

3.3 Software

The software requirement for the control station can be divided into three categories:

i) real-time executive

ii) control based applications software

iii) communications software

The communications software will be discussed in Chapter 6 and the local interface section of the applications programs will be covered in Chapter 5.

The basic functions for a real-time executive for the μC control station are:

i) task initialisation and scheduling or rescheduling

ii) intertask communication and synchronization

iii) timing

iv) interrupt handling

v) input/output handling

In this context I/O handling tends to be very situation specific and covers both process I/O and human I/O. Process I/O includes control of the multiplexor, sample and hold, A/D units and similarly the D/A's or pulse systems on output. Interrupt handling can be effected in hardware if a vectored system is used or in software by a polling routine which, on

receipt of an interrupt, polls each possible device in turn to determine the interrupt source. The timing function must be external to the CPU to ensure some absolute measure of real time. Task initialization is usually based on the timer input as the tasks in question are typically the control routines. Task scheduling or rescheduling may be necessary to alleviate uneven CPU loading. The final function is intertask communication. This may occur within a CPU or if external to (or across) CPU's may make use of the communication system. Because in the process control situation the number of tasks is limited (see Chapter 2), 8 inputs per CPU, the executive in a control station is uncomplicated and consequently reliable, fast and compact.

The applications software is composed of a series of routines which are used to effect the software processing between input and output (Fig. 3.1). The engineer will specify the routines to be used for each input and the list produced, be it a program written in some process control language or literally a list of routine calls, constitutes one task under the executive. A basic collection of programs would cover:

i) a software filter

ii) linearizing and scaling routines

iii) alarm checking

iv) control routines

v) output checking and formatting.

The hardware filter will have removed most higher frequency noise from the signal, but the data may still need to be smoothed. The most common technique is the exponential filter, a simple first order lag.

$$Y_{t_n} = Y_{t_{n-1}} + \alpha(X - Y_{t_{n-1}})$$

where Y_{t_n} = present output

$Y_{t_{n-1}}$ = previous output

X = input

α = filter constant = $\frac{T_S}{T_F}$

T_S = sampling interval

T_F = filter time constant

With α set to one, the output equals the input. As α gets less, so less of the new input and more of the previous output is used, the system becomes more sluggish, until at a value of zero the input is not used at all. The recommended range for α is 0.7 to 0.9. There are other possible filters that introduce less phase lag but the computational requirement becomes too great (3-9).

Linearizing routines cover just about every mathematical possibility. For example, a pH signal may need an inverse sinh function, a flow meter based on pressure drop uses a square root function, and thermocouples may require a high degree polynomial (or large look up tables). Once linearized, the signal is then converted to engineering units by a simple scaling:

$$Y = AX + B$$

where Y = output

X = input

A = scale factor

B = offset

Alarm checking covers absolute alarms, with and without a deadband, offset alarms (relative to some setpoint) or deviation alarms to check rate of change. The deadband is used to ensure noise does not cause multiple alarms when the process variable is hovering on an absolute limit.

Control routines include PID algorithms that output absolute values or incremental changes, that react to some function of the process error,

that have variable gain or may have remote setpoint facility for cascade operation. Refinements include bumpless transfer when changing from manual to automatic operation and back, set point tracking for cascade controllers, and the ability to incorporate interlocks and overrides in the control scheme. Other control routines will be needed for deadtime compensation, feedforward control and derived variable control.

The output checking facility is used to limit the range of the control variable. It is also used to keep the rate of change to a level acceptable for the control element actuator. A side benefit of the rate of change limit is that a CPU malfunction cannot affect the process too quickly, hopefully not before the error has been corrected.

The range and complexity of the software processing options can cause confusion for the process engineer. His method of implementation of these routines has a direct bearing on how they are written for the control station. Chapter 4 covers this problem.

Table 3.2. Final Control Elements

Linear Position Actuators

- Pneumatic
- Electropneumatic
- Hydraulic
- Electrohydraulic
- Electric
- Mechanical

Variable Speed Drives

Variable Electric Power Actuators

Control Valves

REFERENCES

3-1 Chemical Engineers Handbook, 5th Edition, pp22-33.
R.H. Perry, C.H. Chilton, McGraw-Hill, Kogakusha, 1973.

3-2 Ibid. pp.22-87.

3-3 Transducer Interfacing Handbook
D.H. Shingold, Analog Devices, 1980.

3-4 Analog/Digital Conversion
D.H. Shingold, Analog Devices, 1972.

3-5 "Consider Every Error Source for Data Acquisition Design"
D. Chase, Control Engineering, June 1980, pp65.

3-6 "System Architecture Affects Data Acquisition Accuracy and Flexibility"
D. Chase, Control Engineering, July 1980, pp83.

3-7 "Data Acquisition Can Falter Unless Components are Well Understood"
D. Santucci, Electronics, November 13, 1975, pp.114.

3-8 "Manouvering for Top Speed and High Accuracy in Data Acquisition"
D. Santucci, Electronics, November 27, 1975, pp.115.

3-9 Smoothing, Forecasting and Prediction of Discreet Line Series.
R.G. Brown, Prentice Hall, 1963, Section III.

3-10 "Designing and Programming Control Algorithms for DDC Systems"
E.H. Bristol, Control Engineering, January 1977, pp.24.

3-11 Electronic Design's Gold Book, Vol. 3
Electronic Design, 1976-1978.

CHAPTER 4

ENGINEER'S INTERFACE

Figures

4.1 Function

The engineer's interface is his means of communicating with the control system. It differs, in this respect, from the operator's interface which provides a view of the process rather than the control system. The engineer has two primary tasks that he must accomplish through his interface:

i) configuring the control strategy

ii) initializing the parameters in the system.

Modification of the configuration or the control parameters is a secondary requirement but still important.

For the engineer to make optimal use of the control system, the major design consideration should be flexibility. It must be capable of implementing any and all of the engineer's control schemes. The most flexible system available at present is to allow him to program his strategy in a real time computer language. These range from extensions of Fortran and Basic, to specifically designed languages such as RTL and Coral 66 (4-4). The advantage of this flexibility is the use that can be made of process knowledge. Full utilization can be made of feedforward control, derived variable control and signal linearization to improve the quality of control. However, the requirement that the engineer be a capable programmer is the biggest disadvantage with this approach, although modern graduates' experience with computing is fast removing this constraint. An improvement in the situation can be made by providing, as part of the language, a range of standard routines that the engineer would normally require in a control scheme. This would include alarm checking, filtering, PID controller algorithms, and others as mentioned in Chapter 3.

The other end of the spectrum in interface designs is to provide a complete set of standard routines that, in effect, mirror the analog components the engineer would have used. These are then joined using software links in place of wires. This provides a very simple interface,

but the tradeoff has been at the cost of flexibility. It becomes difficult to provide those blocks that direct programming can account for, without increasing the number of available blocks to the detriment of simplicity. A system that uses the block approach is ABACUS (4-3).

To choose between these two extremes, we must examine the engineer's approach to control system design. Typically, a control strategy is first committed to paper as an instrumentation diagram, a graphical layout. The problem arises when an analysis is done of the requirement of each component in this diagram. If a block approach was adopted, what choice of standard blocks should be made available to ensure adequate flexibility, without over-complicating the interface. Harris and Bell (4-1) suggest that the 22 signal processing functions of the SAMA RC 22-11-66 standard would be suitable. Perhaps the best criteria concept for deciding is that of information chunking (4-5). Briefly, the more information contained in each symbol, the fewer symbols are needed to express an idea. For example, using the decimal number system where each symbol represents a choice of one of ten, the number 197 requires three symbols. In the binary number system it needs eight, 11000101, and in hexadecimal only two, C5. So while the 22 SAMA functions may be adequate, to improve flexibility, it may be necessary to provide several refinements of one function. Leeds and Northrup do this in the MAX1 system by offering six variations of the basic PID algorithm (4-2). For the engineer to be totally familiar with the possibilities of such a system, and thereby be able to design any control strategy he wishes, is becoming not much simpler than direct programming.

Another problem is providing, within the block algebra structure, the capability of modifying any parameter, based on another input or the output of some other block. Again, this is easily accomplished with a real time language.

To reduce the mental jump from the graphical layout to the computer implementation, therefore, a combination of the two approaches would seem best. Give the engineer the facility to represent the control scheme as a series of functional blocks, but give him the flexibility to implement a function of his own choosing within each block, if a suitable one does not already exist.

The initialisation process can be completed at the same time as the configuration is done, or it may be left so as to take advantage of a data file that may be stored within the computer.

4.2 Software

The information input by the engineer represents a set of files, one per process input, that each contain a set of records, one per block. The total is, in fact, a data base in the classical commercial sense. The engineer's interface, therefore, represents one part of a data base management system (DBMS). The design of a DBMS will then provide a guide-line for the software required in the interface. The DBMS is more fully covered in Chapter 7 but the relevant concepts will be stated here.

Typically there are two parts to DBMS interface software:

i) a data definition language

and ii) a data manipulation language.

The data definition language is used to construct the logical storage structure of the data (as opposed to the physical). This has already been set by the method chosen for configuration, one file per input that contains a header record and one record per block. The data manipulation language is specifically what the engineer's interface software represents. The language must do three things; create, modify and delete files. In view of the initial graphical layout of the control system design, this

is best done graphically. Using a touch sensitive graphics screen, a light pen or suitably interactive system, the engineer could select a block, represented graphically by its standard symbol, and place it within the structure as per the instrumentation diagram. Connections, parameters, and functions if required, could be filled in as this was done. The display could be as in Figure 4.1. Modifications and deletions could be similarly handled. With only an alphanumeric VDU the configuration and initialisation could be done in an on-line conversational manner, using menu driven systems as per modern commercial DBMS practise, with preformatted displays.

4.3 Hardware

From the previous discussion, the hardware requirement is:

i) a display unit

ii) an input system

iii) a back-up storage medium.

The display unit could be either a graphics terminal (high resolution is not essential) or an alphanumeric VDU. With the graphics unit, some sort of interactive input system is desirable, for example a touch sensitive screen, and both display media require a standard keyboard for parameter initialisation.

The back-up storage medium is needed to obviate constant data re-entry. After a control station failure, the required information could be taken off the back-up rather than re-entered by hand. Its possibilities range from the expensive Winchester disc system, through bubble memory cassettes to floppy discs or magnetic stripes.

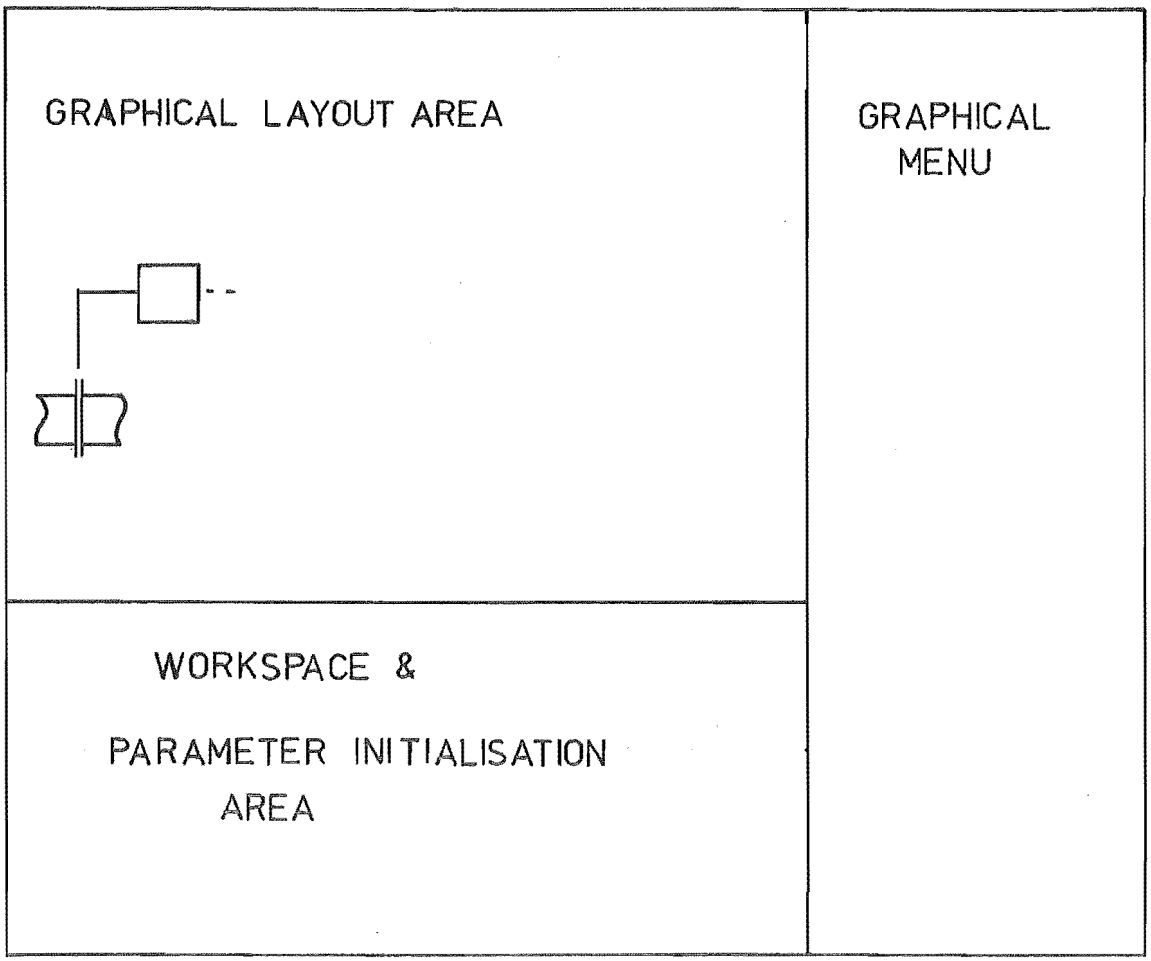


FIG 4.1 ENGINEER'S INTERFACE GRAPHICAL DISPLAY

REFERENCES

4-1 "Symbolic Function-Oriented Programming for Process Control Has Simple Rules"

L. Harris, T. Bill, Control Engineering February 1979, pp.91.

4-2 MAX1 Controller Functional Description

Leeds & Northrup, 1980.

4-3 "Programming by Block Diagrams - a computer language to suit the process engineer"

C.A.J. Payne, Canadian Control & Instrumentation, Dec. 1974, pp25.

4-4 Proceedings of the Conference on Software in Process Control

IEE Conference Publications No. 102, 1976.

CHAPTER 5

OPERATOR INTERFACE

Figures

5.1 INTRODUCTION

The Purdue Workshop Guideline (5-1) outlines a four phase approach to the design of man-machine interfaces:

i) identifying requirements;

ii) allocate functions;

iii) analyze tasks to perform functions;

iv) produce functional design.

Identifying the requirements of the operator interface involves determining what the combination of interface and operator should achieve within a computer based process control system. These requirements will change with developments in the areas of computer technology, control theory and human psychology. Within the framework of current knowledge, however, those requirements that can be laid down give rise to a broad specification for the operator's man-machine interface.

The allocation of functions, that have been identified, between man and machine and the analysis of the tasks that make up those functions is an application of the science of ergonomics. Ergonomics or human factors engineering, as mentioned in Chapter 1, is the designing of work situations to human capability constraints. By knowing the relative capabilities of man and machine, one can produce the best possible division of responsibility between the two parts of the system.

Having completed the task analysis and allocated functions to both man and machine, it is then possible to produce a functional design.

5.2 FUNCTION

Whereas the engineer's interface was the means to specify the control strategy, the operator, through his interface, is the control tactician. He is responsible for the immediate state of the process. His primary task is to keep the process at its optimum and his interface must be designed to help him achieve this. Even with an optimal interface, the operator's model of the process, and thereby his concept of optimum, may prevent him from attaining this goal.

In any automatic system, the man's role is:

i) start up and shut down;

ii) operating and monitoring;

iii) maintenance.

In a process control system in particular, the operator's main responsibility is operating and monitoring. Control system start up and shut down, including installation, wiring and initializing is, foremost, the control engineer's domain. Instrumentation engineers, typically, look after maintenance. Operating and monitoring of the process and control system can be further subdivided:

i) process start up and shut down and other sequential tasks;

ii) normal control, which involves stabilization
optimization
and breakdown avoidance;

iii) abnormal control, which involves
detection of an abnormal condition
diagnosis of fault
correction of fault condition;

iv) data logging and recording.

These functions must be divided between the operator and his interface so as to optimize overall performance. This places a critical responsibility on the design of the interface because the system designer has little or no control over the characteristics of the operator. He may be able to specify that the operator not suffer from colour blindness, deafness or similar handicap, but his design should be able to cope with a wide variability in the physical and psychological makeup of human operators.

The Guideline (5-1), in the introduction, states the interface should be considered as a translator. Operator inputs are converted into process variable changes and process data is translated into displays for human comprehension. There are four separate functions, each posing different problems for data translation, and the interface must be able to handle each.

5.3 ERGONOMICS

As a result of applying ergonomics to the design of the operator interface, it should support the operator in every facet of his job. To be able to do this, we must first consider the relative abilities of men and machines. Table 5.1 gives a list of relevant characteristics. On the basis of this table we can allocate some tasks to man or machine directly and, in other cases, determine where the machine is best able to support the man.

Looking at the list of functions in the previous section, we can first look at data logging. This involves repetitive recording of data, with some processing. The machine is capable of handling this task with no intervention required on the part of the operator.

Table 5.1: Man/Machine Capabilities

	MAN	MACHINE
Power:	Low power output for long periods. Output is a function of time.	Large outputs for long periods, independent of time
Data Processing:	Poor data processing ability, but versatile. Can handle novel situations. Inductive reasoning.	Fast, accurate and can handle complex simultaneous functions. No intelligence, deductive reasoning only.
Speed:	Slow with lags. Limits on data transmission rate.	Fast, repetitive.
Signal Processing:	Can handle a wide range of stimulae. Interpretation and extrapolation. Perception of space and pattern. Noise tolerance.	Good for signal outside human range, absolute judgement and accurate measurement. Can integrate and differentiate signals.
Memory Capacity:	Large long-term memory, small short-term.	Large fast short term store. Relatively small long-term memory.
Failure Mode:	Gradual degradation of performance due to stress, fatigue.	Sudden breakdown. Relatively constant short term performance.

The next task to consider is process start up and shut down. Typically this involves a series of sequential steps, each not being started until the previous one has finished. For each a large number of possible error conditions can exist and it is very unlikely that the engineer can 'a priori' program a correction for every one of these. The operator's adaptability is therefore needed. The computer can, at best, monitor the plant and report the completion or non-completion of each step. The operator can then either correct the fault or initiate the next step. The most suitable sort of display to use, to present the information to the operator, will depend on plant size. A small plant may make use of a pictorial layout diagram without overwhelming the operator. A large plant, however, will be better off with a less graphic and more tabular presentation.

The last two functions are normal and abnormal control. The most important of these is abnormal control, as it is under this situation the operator suffers most stress. If his interface is not designed to provide maximum help in this circumstance, the user will tend to reject it completely. The three steps, for the operator, under abnormal control are:

i) detection;

ii) diagnosis;

iii) correction.

The computer can aid in the detection of an abnormal condition by comparing process variables against set alarm limits. The operator will be notified if a limit is violated. To notify the operator, use of as many human input channels as possible is preferred. Typically this is done by using several different visual cues and an audible signal. Visual cues vary from a change of size or colour to using blinking indicators. The use of this redundant coding reduces the chance of a fault going unnoticed through inattention or being lost in other information.

Typically process alarms do not occur singly. Normally one exceeded constraint in the process will result in many others downstream also being violated. The interface must prevent the operator from being overloaded by repressing these secondary alarms and concentrating the operator's attention on the primary cause. This is not always easily achieved as the primary alarm is not always the first alarm in a sequence.

The abnormal condition has now been detected and the operator's next task is diagnosis. Here, the interface designer should make use of two basic principles. 'Management by exception' means giving the operator only that information in which he is interested. The second is the use of a hierarchical diagnostic search. To find the cause of an alarm, the operator must first locate the source of the fault and then determine its nature. If the operator was permitted a random search through all loops, his diagnosis would be time-consuming and very probably inaccurate. It is more efficient to give him an overview of the plant with the information collected into small subunits. Determining which subgroup is responsible for the fault, he is then presented with more detailed facts on the loops within this subgroup. Should he then deem it necessary, he is able to select one loop and get a completely detailed display of data pertaining thereto. This forced search prevents preconceptions, on the operator's part, from allowing him to immediately concentrate on what may be the wrong loop. A familiar example of this problem is the car with an empty petrol tank and an owner busy under the bonnet searching for the fault. The division into only three levels is also very deliberate. Because of the operator's limited short term memory, any more levels in the hierarchy can result in the operator forgetting his whereabouts in relation to the whole plant. During fault location, the operator is only check reading. He is making a qualitative comparison to see if a variable has exceeded specified limits. Determining the nature of the fault requires some historical data, again only qualitative. Because of man's pattern recognition ability, this

information is best presented graphically. A suitable symbol to represent each subgroup within the overview display and another for each loop within a subgroup display could be used to present this information. The choice of symbol should be based on the following perceptual principles (5-7).

i) figure versus background

ii) figure boundaries

iii) closure

iv) simplicity

v) unity

Another point, not mentioned by McCormick, is that the symbol should have some relevance to the user. For example, in a process control situation the loop symbol should be reminiscent of an analog controller display to take advantage of operator experience. In an intensive care nursing situation, a relevant overview symbol for each patient could show a human face whose expression changes with the patient's condition. Briefly explaining each of the above points; figure versus background means that the symbol should not get lost in background clutter, the figure should have well defined limits to confine the area of attention, the figure should be closed in that all data should be within these limits, it should be as simple as possible commensurate with the information it needs to convey and lastly, it must have a sense of unity or completeness. A possible subgroup symbol is shown in figure 5.1 and a loop symbol in figure 5.2. Presentation of historical data is best done through a trend graph (see figure 5.3). It is not necessary to provide detailed scales as its purpose is only to show a trend. The time scale, however, should be variable from as short as the last five minutes up to the whole of the previous day. If a separate trend display is included in the interface, it gives the operator the chance to correlate two process variables with one another.

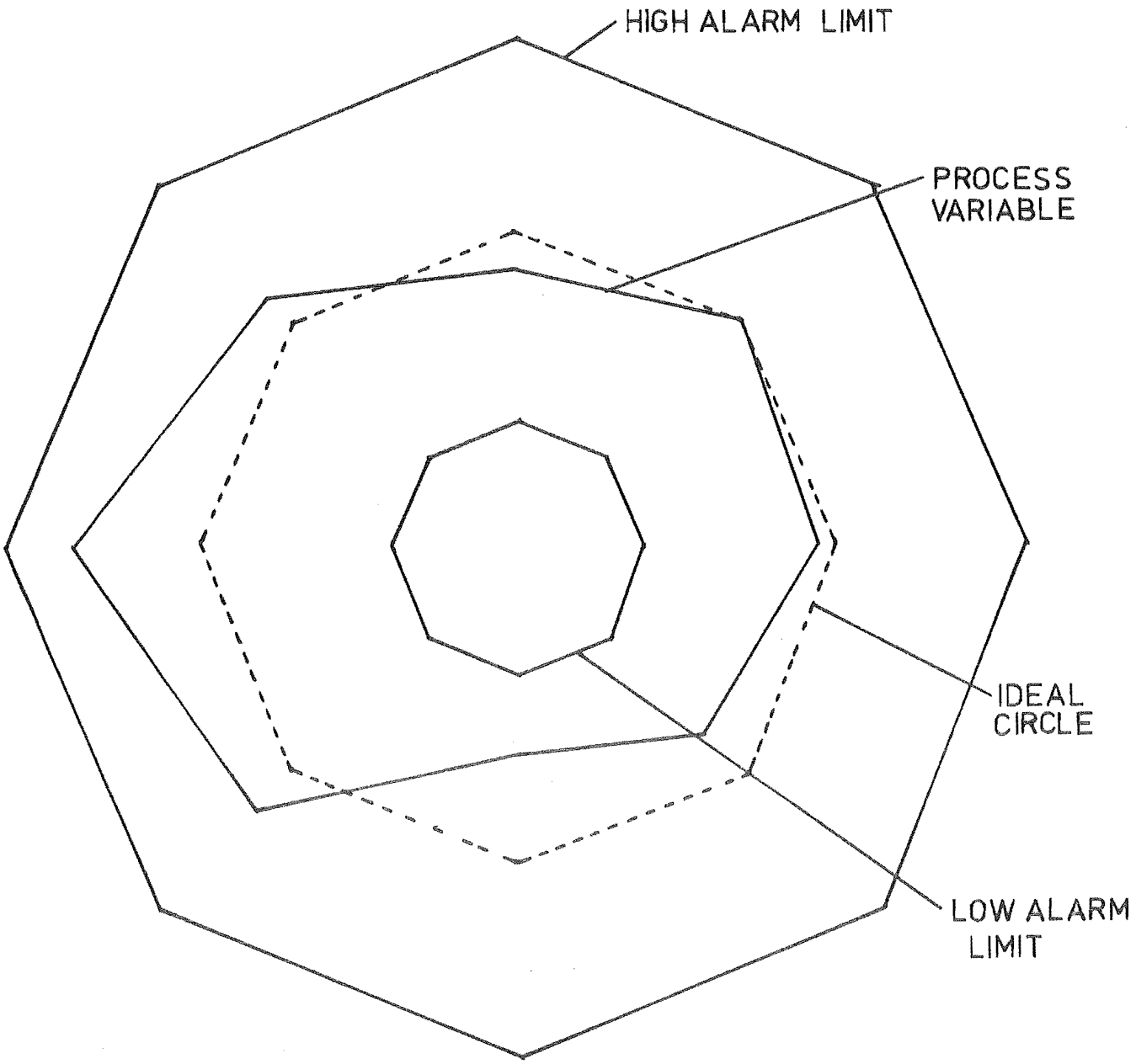


FIG 5.1 SUBGROUP SYMBOL

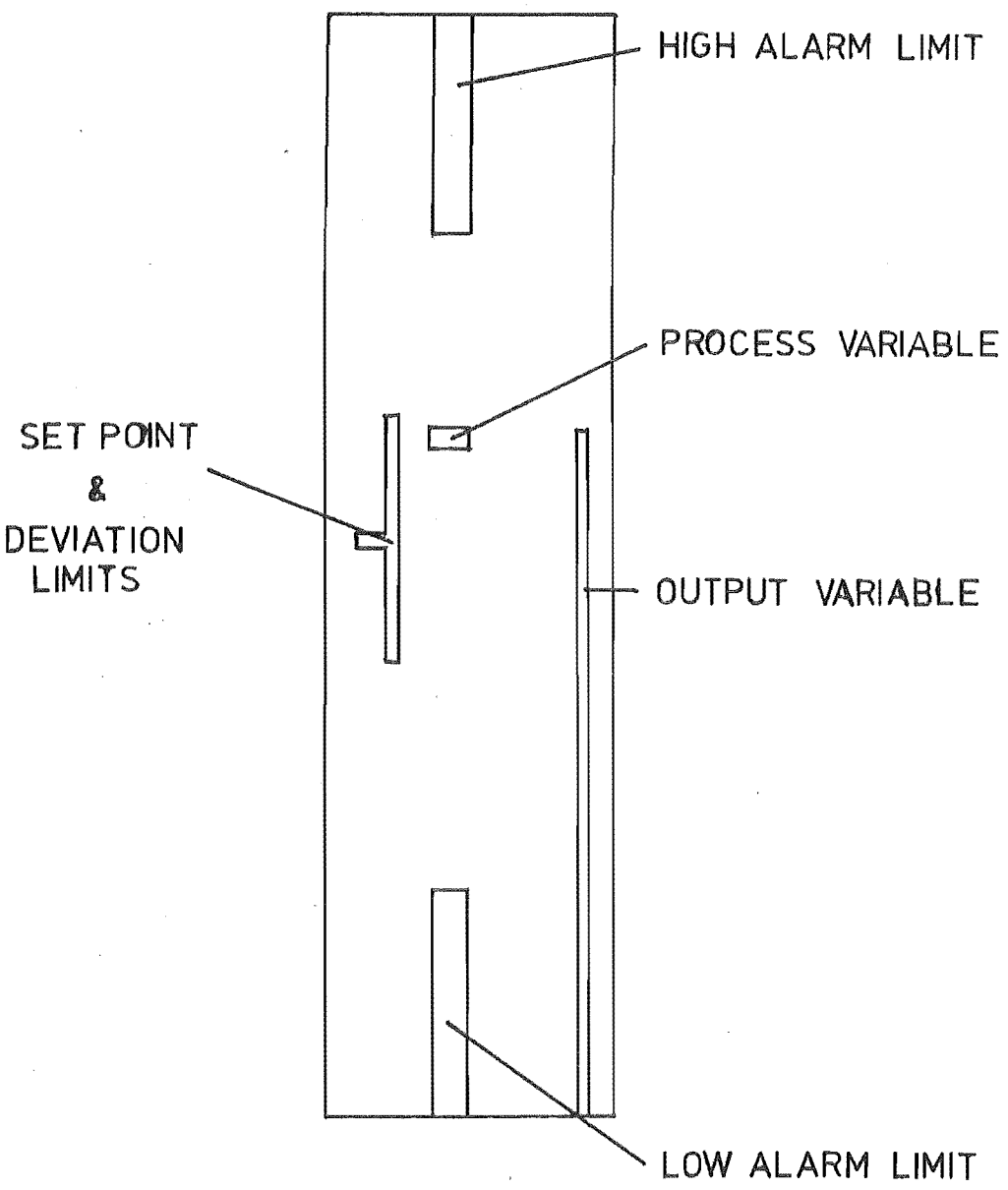


FIG 5.2 LOOP SYMBOL

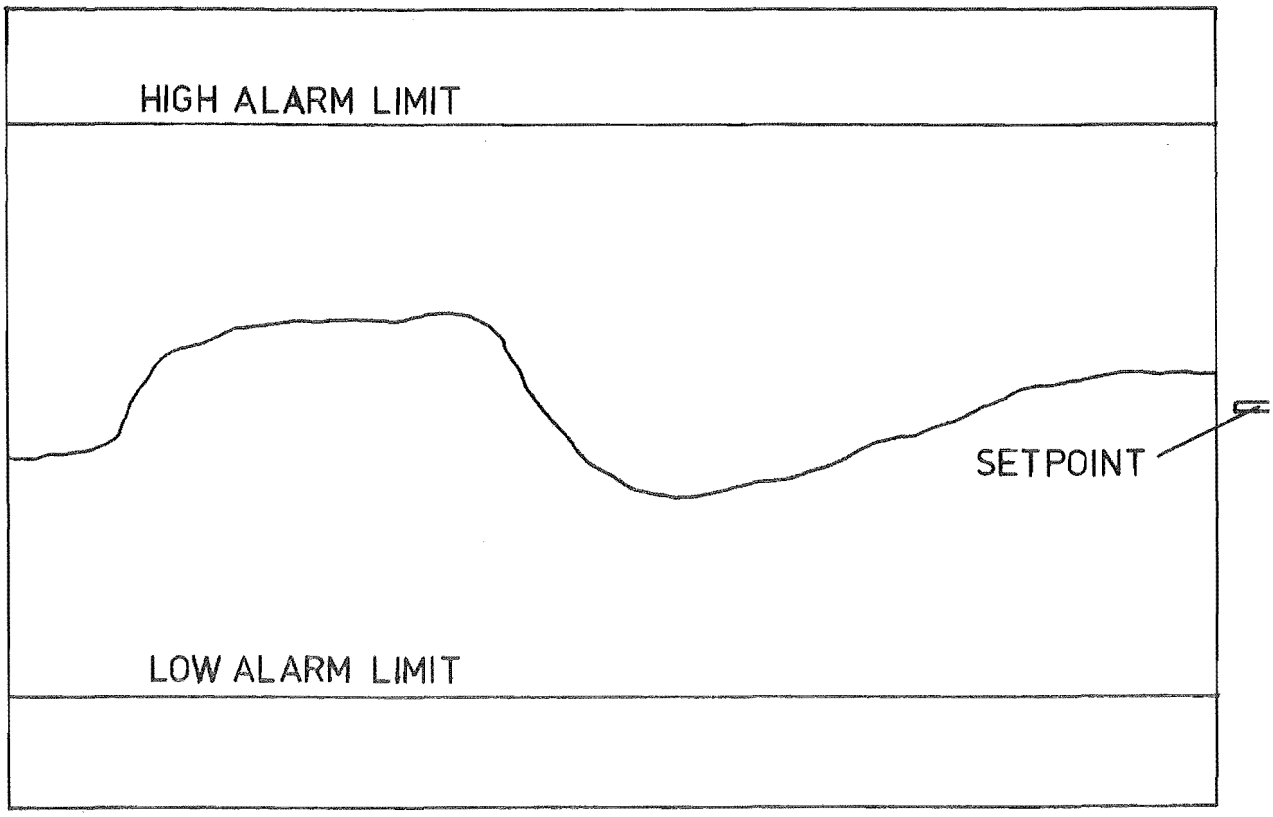


FIG 5.3 LOOP TREND GRAPH

The previous two steps were mainly concerned with information presentation. The last step, fault correction, involves the input of data. The operator has the following possible inputs:

i) selection of - a subgroup display
- a loop display
- a variable to be altered
- the overview display

ii) altering - the status of a loop
- the value of a variable.

The first range of inputs are selections between choices already displayed. The simplest method is to have the operator point at what interests him. Pointing requires no training, no skill and is an immediate response. By using a touch sensitive screen most operator inputs can avoid the use of a keyboard. When the standard QUERTY keyboard is considered, this is a distinct advantage as the operators do not need to be expert, or experienced, typists. However, the consideration of error recovery when the operator makes an incorrect selection requires that there be at least some keyboard inputs. Another consideration which suggests the use of at least a minimal keyboard is the 'location coding' of critical functions in the operator's mind. One such function is selection of the overview display under alarm conditions. The operator's response is almost instinctively fast if he does not need to think about the location of the response button. Error recovery is most easily dealt with before it becomes necessary. By making any critical functions multistep, and giving the operator the chance to opt out at any step, the possibility of a catastrophic error is significantly reduced. The last input option, altering the value of a variable, is also best effected through a keyboard. Two methods of doing this are:

i) a numeric keypad, if values are to be entered from scratch;

ii) a set of change buttons, if values are only to be altered by small amounts.

The change buttons can allow the operator to increase or decrease a variable either slowly or rapidly.

Normal control involves optimizing and stabilizing the process, and avoiding an abnormal condition. Optimizing, for the operator, means comparing the present state of a variable with its desired state and reconciling the two. It also means exercising some control over the path the variable will take to achieve its desired value. Control of the transition path also involves consideration of the stability of the process. It may be possible to force an individual variable directly to its desired state, but this may result in instability elsewhere in the process. To achieve an accurate comparison the operator needs to make a quantitative check on the variable being studied. This needs digital data.

Avoidance of an abnormal condition depends on the operator's experience and his mental model of the process. This mental model is a direct result of the interface design. It can vary from a physically real model, if mimic displays are used, to a more abstract, but no less effective, relationship between input and symbol shape change. The trend graph mentioned earlier also plays a significant part in breakdown avoidance by providing a dynamic view of the process that also contains historical data. The subgroup symbol and the loop symbol can provide some dynamic indication by displaying previous values at the same time as current ones, but eventually clutter destroys the symbol's effectiveness.

The next point that should be made is that under normal control conditions the operator will be checking related loops within and across subgroup boundaries. Some means of accessing these other loops without having to repeatedly traverse the complete hierarchy should be provided. For example, three similar reactors within a plant may be represented as separate subgroups in the interface. The operator may wish to study all the loops related to one reactor, or similar variables in each reactor.

The tedium of having to return to the subgroup or overview display would cause him to reject the interface.

The last ergonomic points about an operator interface design are those of training, confidence and pacing. The last two are somewhat inter-related. Pacing, in this context, means who the operator perceives as being in control, himself or the interface. If the machine paces the man, the machine is telling the man what to do. This will adversely affect operator confidence, as nobody likes to be dictated to by a machine! The man should always pace the machine, or at least think he does. To avoid this, the interface should never tell the operator that his proposed course of action is not permitted. Neither should the interface tell the operator what to do. All operator actions should illicit a response and the operator should be able to do as he chooses. The other side to operator confidence is ensuring the interface is always appearing to do something. For instance, the time of day may be displayed, being updated every second. If the interface is busy processing a previous request, this fact should be made plain to the operator. If the interface is apparently idle or does not respond to a request, the operator may assume a fault has occurred and reject the interface.

In considering operator confidence, one more display is essential to the interface. A hardware status display is needed to keep the operator informed about the state of all pieces of hardware in the control system. The use of a display like this means that at all times he is aware of what the control system is not doing and why not.

Training requirements should be minimal. The less training is needed, the fewer errors an operator is likely to make and the more confidence he will develop in the interface. There is an additional piece of hardware or an additional display that will be needed for inexperienced operators

and that will help reduce training time. A pictorial representation of the process showing where each control loop is will help new operators realize cause and effect of alarms and actions. It is also an advantage when the engineer, for example, is called in and is not totally familiar with control loop representation in the interface. Rather than use a display, it is better to mount the flow diagram on a wall behind the interface so it is always available and can be referred to without losing information currently being displayed.

5.4 FUNCTIONAL DESIGN

The discussion on ergonomics has resulted in a generalised outline of the operator's interface. In this section the means of achieving those goals will be suggested.

There are two components to the interface:

i) the display system

ii) the input system

The display system needs to be dynamically reconfigurable, to allow for the variety of displays, and also to allow for changes in the process and control system. Other constraints are that it be capable of real time graphics, preferably in colour to make use of redundant coding. The best means of achieving this ideal currently, is with a colour graphics CRT terminal. This allows the display to be software formatted and is capable of real-time change.

The input system can be further subdivided, according to the previous discussion:

i) a keyboard

ii) a touch sensitive system for the display

The keyboard should be designed with coloured, backlighted pushbuttons, or a system capable of achieving the same result. The colour enables the buttons to be grouped by function according to location and colour. This reduces the likelihood of errors and cuts down on training time. The use of backlighting can help to show the operator that a button has reacted to his touch, increasing his confidence, or it can be used to show active and inactive functions. A list of the keyboard functions needed to satisfy the requirements of section 5.3 are:

i) Alarm Acknowledge, to turn off the audible alarm signal.

ii) Clear Display (or Overview Display), to call up the overview display.

iii) A Clear and Enter combination, to allow error recovery.

iv) A set of buttons to change variable values.

v) A set of loop addressing buttons for optimization of related loops.

This list provides a vast reduction on the number of keys in a standard QUERTY keyboard, resulting in a simple interface design.

The touch pad system can use one of four approaches:

i) a conductive membrane, providing a voltage divider effect;

ii) capacitance;

iii) acoustic detection using a reflected sound signal (similar to radar);

iv) an infrared LED/photodetection light bean grid.

The most versatile and reliable of these is the infrared light beam array (5-6) An X and Y axis array will give the location of a pointer when both beams are broken. Whereas the backlight was used to provide a feedback signal to the operator with the buttons, the touch pad will need an audible signal. A light signal will be difficult to detect near the display unit.

Before implementing the design, the systems engineer must produce a state diagram to enable the sequential operation of the interface to be specified. An example of one is shown in Harison's Chapter 7 (1-1). The state diagram should include the allowed transitions from display to display and the necessary steps that the operator must take to make these transitions.

REFERENCES

5-1 Guidelines for the Design of Man/Machine Interfaces for Process Control
R.F. Carroll, Purdue University, June 1976.

5-2 Man and Computer in Process Control
E. Edwards, F.P. Lees, I.Chem.E., 1972.

5-3 The Human Operator in Process Control
E. Edwards, F.P. Lees, Taylor and Francis, 1974.

5-4 Ergonomics
K.F.H. Murrell, Chapman and Hall, 1965.

5-5 Information Transmission
E. Edwards, Chapman and Hall, 1964.

5-6 "Touch-Sensitive CRT screens join computers and non users"
Electronic Design, Oct. 15, 1981, pp.61.

5-7 Human Factors in Engineering and Design
E.J. McCormick, 4th Edition McGraw-Hill, 1976.

CHAPTER 6

COMMUNICATION SYSTEM

Figures

6.1 Function

As was mentioned in section 2.1, there are three functional parts of a computer-based process control system:

i) storage

ii) presentation

iii) communication

The communication subsystem becomes a major part of the total system when the computer intelligence is distributed. It must effect the following data transfers:

i) storage-storage

ii) storage-presentation

iii) presentation-storage

where these sections are in separate machines. To do this requires a software interface at each end, a hardware interface and the transmission medium (Fig. 6.1).

6.2 Communication Technology

Digital communications systems can be analysed on three levels:

i) network topology

ii) hardware

iii) software

The last two are frequently combined into a network protocol. A protocol in this sense is a set of rules that define the communications system. As the technology has grown, the protocols have needed to be able to accomodate the many diverse applications of digital communications. This has resulted in a multi-level protocol definition. The most complete is the protocol designed by the Consultative Committee for International Telephany and Telegraphy (CCITT). It is a seven layer protocol:

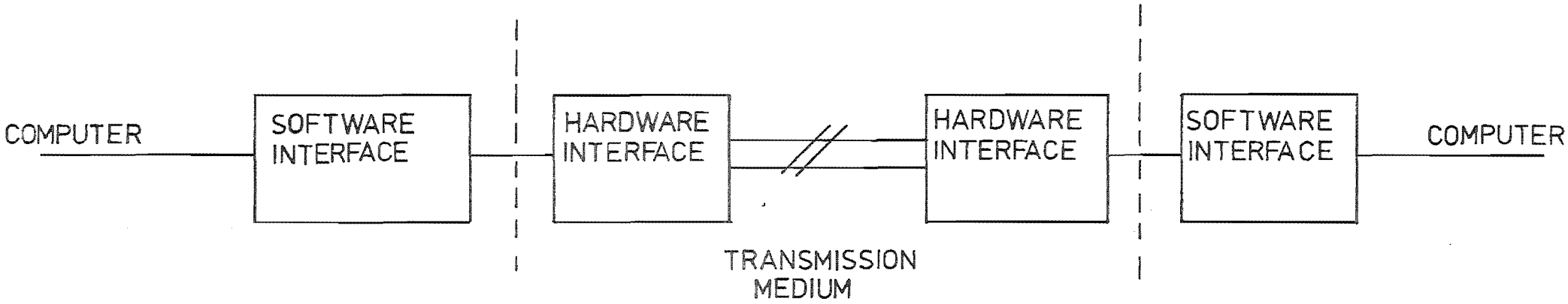


FIG 6.1 BASIC COMMUNICATION SYSTEM

i) Physical control

ii) Link control

iii) Network control

iv) Transport control (end-to-end)

v) Session control

vi) Presentation control

vii) System control

Each layer is independent of those above and below so an improvement can be made in any layer without affecting others. Also, each layer is transparent to those above, so, effectively, communication is between equal levels in different computers (see Fig. 6.2).

Those levels of most interest for process control systems are the first two. Level 1, the physical level, covers both the mechanical and electrical specification. The transmission media most commonly used as the data highway are twisted pairs or coaxial cable. Optical fibre, while having many advantages, particularly immunity to electrical noise and the provision of electrical isolation, suffers because of the mechanical difficulty of optical connection. The choice between a twisted pair and coaxial cable is a trade-off between cost and bandwidth. (It is also for these reasons that high bandwidth parallel data highways are not used over the long distances involved in process control systems). A twisted pair can handle frequencies up to about 200 kHz and a coaxial cable up to 10 MHz but the useable speed of transmission is affected by the distance to be transmitted. Kelvin's Law states that the maximum operating speed is inversely proportional to CRL^2 where C is capacitance per unit length, R is resistance per unit length and L is the length (6-4). The method of transmission over this media must also be considered here. A carrier signal can be modulated with the data or the data can be sent direct. Modulation requires the use of machines at receiver and transmitter.

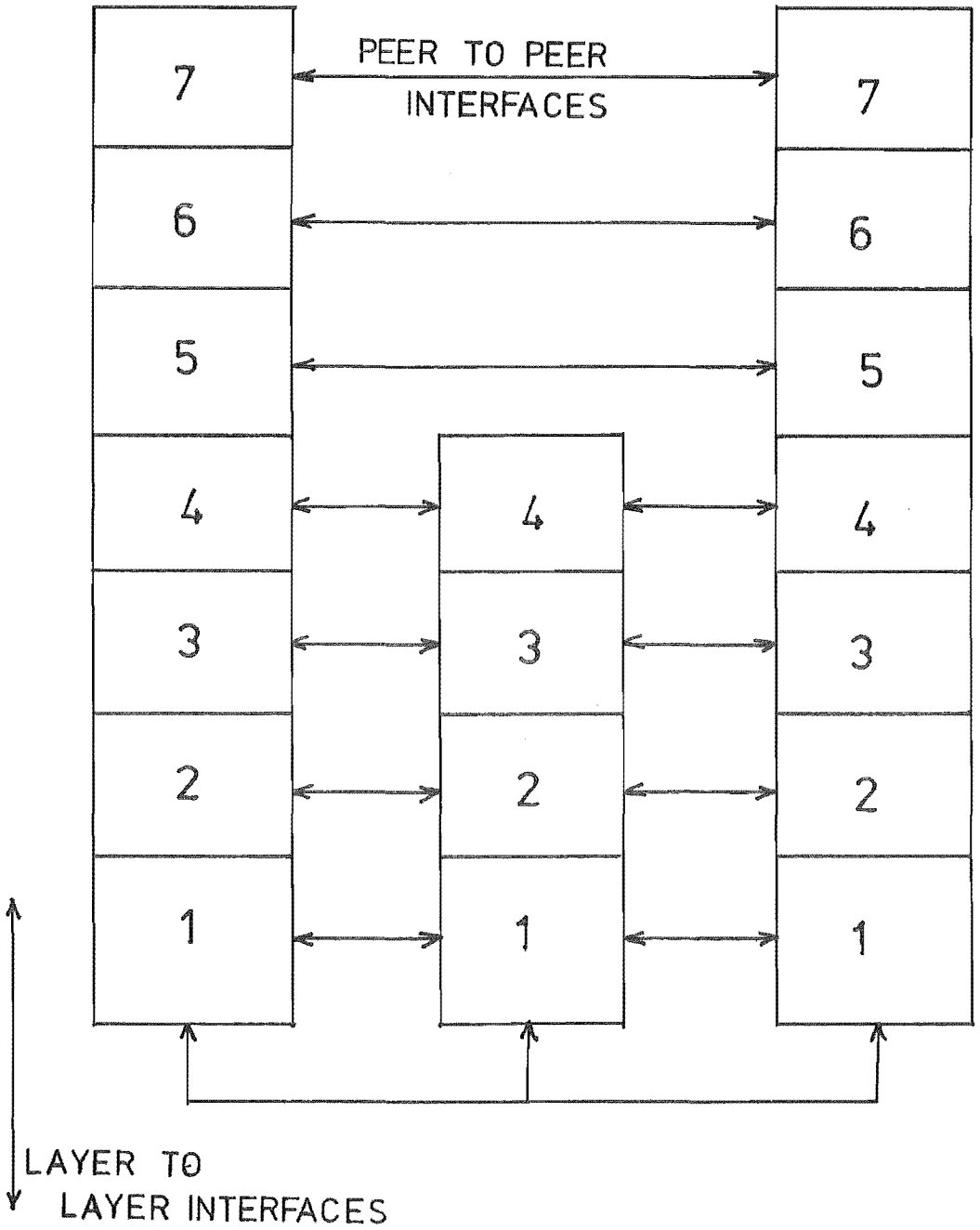


FIG 6.2 LAYERED PROTOCOL STRUCTURE

The type of modulation can be amplitude, frequency or phase or any combination of these. The advantage of a modem is the capability of two-way simultaneous transmission on different frequency carriers. Direct, or baseload, transmission uses a variety of codes (Fig. 6.3):

i) Pulse - a pulse = 1, no pulse = 0

ii) RZ - return to zero; a +ve pulse = 1, a -ve pulse = 0

iii) NRZ - non-return to zero; a +ve transition = 1, a negative transition = 0

iv) NRZI - non-return to zero inverted; a transition = 1, no transition = 0

v) Manchester - a -ve transition at data time = 1,
a +ve transition at data time = 0,
other transitions are ignored.

vi) Harvard - two pulses in the same direction = 1, in opposite directions = 0.

Codes ii) to vi) have the advantage of an average DC value on the line of zero and codes i), v) and vi) have a pulse at every clock interval and therefore can be used by synchronous transmission without an extra clock signal. Code i) is used in the common 20 mA current loop. Several standards exist to cover the electrical interface between level 2 and the transmission medium. The more common ones are RS-423 and RS-422 (or their CCITT equivalent V.10 and V.11), HPIB (also known as IEEE 488 and numerous other commercial names) and CAMAC.

Level 2 is the link control. Link control protocol are either asynchronous, each character having a start and stop bit for timing, or synchronous where the receiver and transmitter must be in synchronization, through the use of a clock signal (see the discussion on codes above). Synchronous protocol are classified into three types depending on the message framing format used:

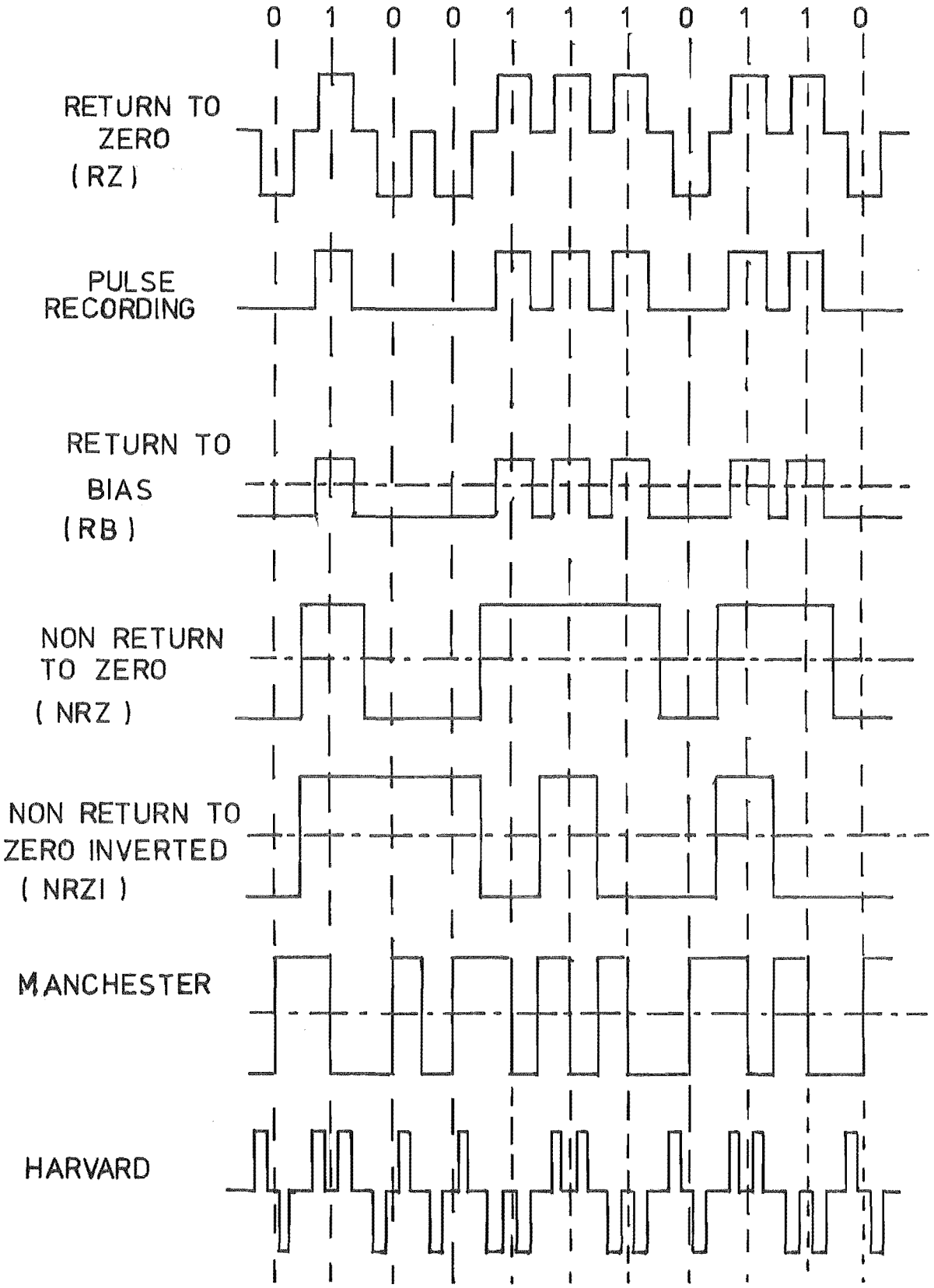


FIG 6.3 TRANSMISSION CODES

i) Character oriented protocol

ii) Byte oriented protocol

iii) Bit oriented protocol

Examples of each are:

i) Bi-sync (IBM)

ii) DDCMP (DEC)

iii) SDLC (IBM), ADCCP (ANSI), HDLC (ISO), X.25 (CCITT), CDCCP (CDC), BDLC (Burroughs)

A comparison of the features of these protocol and a list of commercially available devices capable of implementing them is listed in Weitzman (6-5) pp.380-381. The important features are (6-6):

i) Initialization

ii) Framing

iii) Link management

iv) Error control

v) Sequence control

vi) Flow control

vii) Transparency

viii) Abnormal condition recovery

Initialization involves the setting up of a data link, activating receiver and transmitter and, if necessary, synchronizing the two. The three different methods of framing are start and stop characters or bits, byte count or a flag at the beginning and end. Link management includes initialization and in addition makes decisions such as which will be active on a multidrop line. More link management is effected by levels three and four. Detection, correction of errors and acknowledgement of correct transmission is the responsibility of the error control section. Detection uses vertical or longitudinal redundancy, or parity as it is more commonly

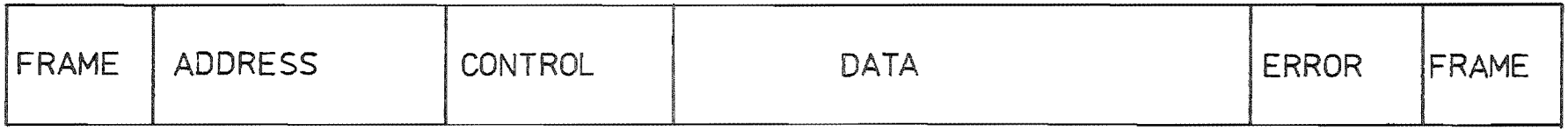


FIG 6.4 MESSAGE PACKET FORMAT

known, or the more efficient cyclic redundancy check (CRC). CRC is the process of dividing the message block by a given number, and the remainder, hopefully unique (this depends on the choice of divisor), is the error check code. CRC is preferable because it is capable of detecting burst mode errors, the most common in process control situations. Error correction is achieved by a repeat transmission of the message. This is also preferred because of burst mode errors which may have affected the error code, preventing the use of error correcting codes. Sequence and flow control, some of which is also done by level three and four, looks after the numbering of transmission to detect missing and duplicate messages and to handle the buffering of messages until an acknowledge is received. The number of messages that may be stored depends on the type of retransmission-request scheme being used (6-7). The two most common are 'stop and wait', where each message must be acknowledged before the next is sent, and the 'go back N' types, where if a block is rejected either just it is retransmitted or all blocks from there on are retransmitted. It is called 'go back' because transmission is not halted waiting for each block's acknowledgement. Transparency, as mentioned in the initial discussion on protocols, means that the data stream is unaffected by the level two processes. Abnormal condition recovery is the action required when the transmitter detects an illegal situation on the line. These can be non-valid protocols, missing responses or time out caused by link failure. This may involve aborting the current message and returning to the idle state or reinitializing the transmission line, or selecting a different route.

Level three, the network control, provides the functions for message routing over a network and the breaking up of long messages into separate packets.

Level four is responsible for the overall control of a source to

destination link even if this involves several intermediates in a network.

The remaining levels may do things such as data encryption, computation and formatting for a peripheral device on a hardware sharing network.

Apart from the network protocol, we must also discuss the network topology. As mentioned in Chapter two, there are three topologies of interest (Fig. 6.5):

i) star

ii) multidrop line

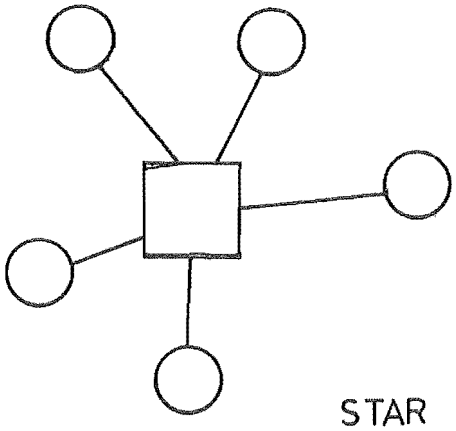
iii) ring

For a full discussion of all possible distributed computer system layouts see Weitzman (6-5).

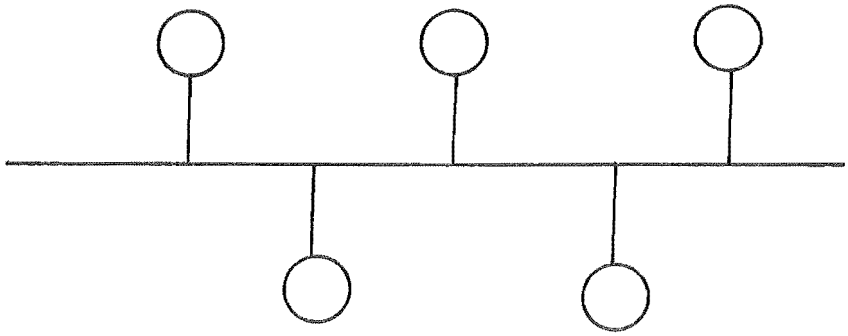
These topologies should be analysed from the viewpoints of cost, performance and reliability.

The star arrangement is the easiest to implement, the complexity being confined to the bus master/switch. Additions to the system are simple until the master capacity is exceeded. It is a reliable system except for the bus master, the failure of which destroys all communications. Reliability is increased by adding duplicate masters. Performance is dependent upon the speed of master and it is here, also, that most of the cost of this layout lies.

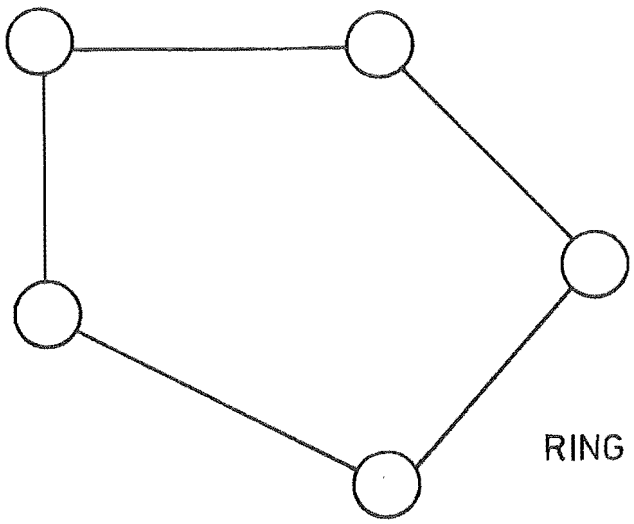
The multidrop line is also conceptually simple. One unit can talk at one time, only, so performance is lower than the star configuration. It is, however, cheaper. There is no need for a bus master, as such. Control of the bus is given to the control station that is transmitting at the time and the next transmitter can be decided by contention or polling. Contention is a case of talk when you have something to say and hope you



STAR

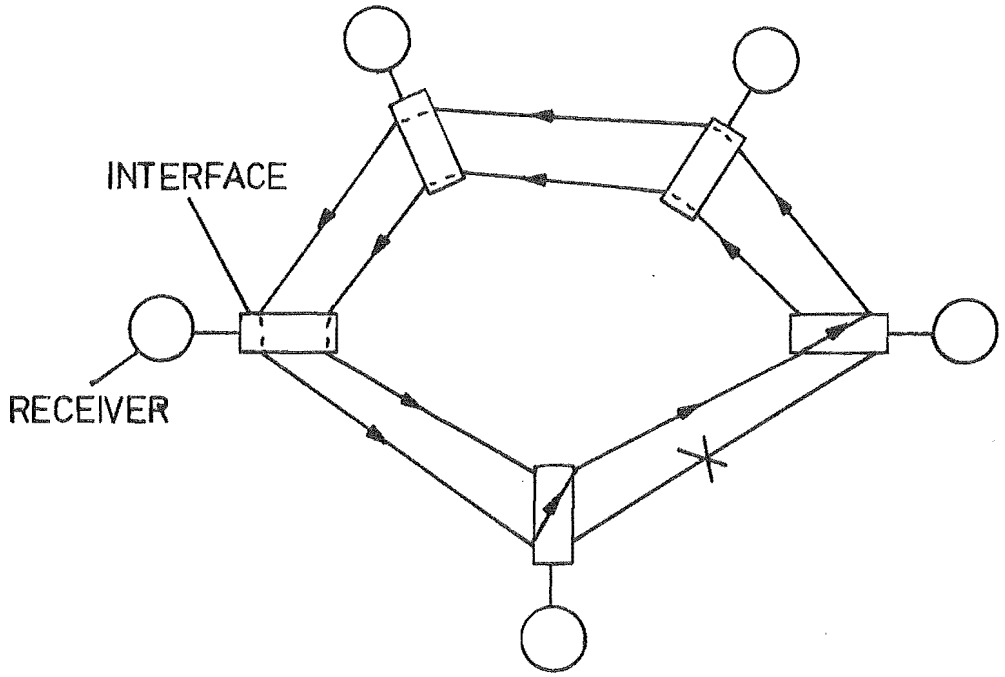


MULTIDROP LINE

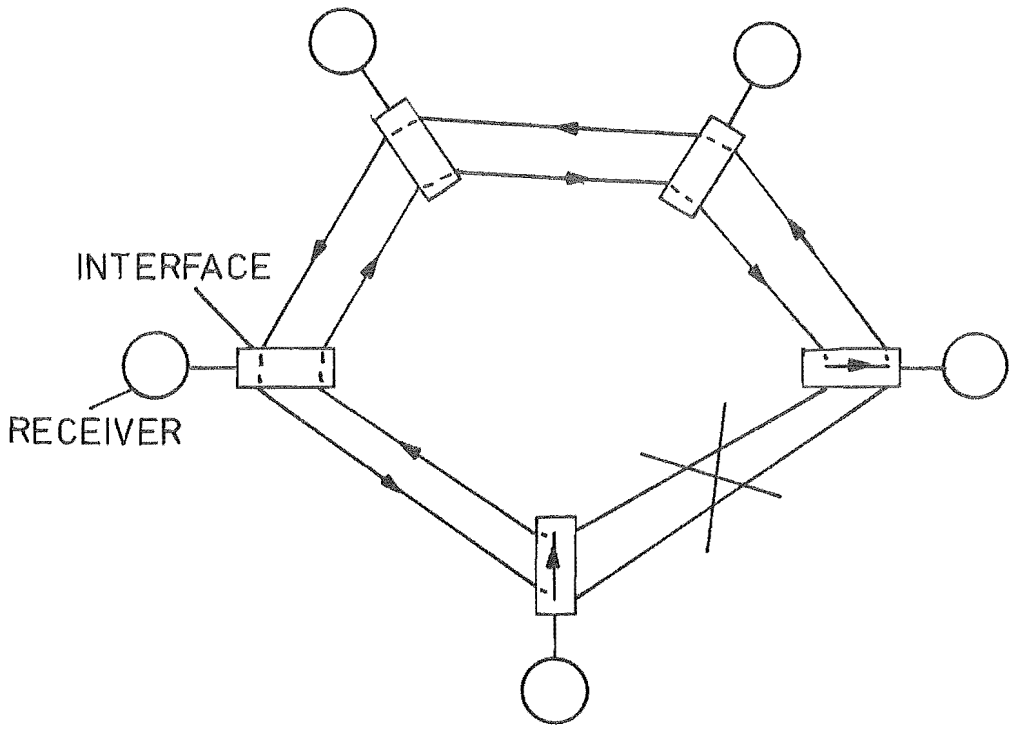


RING

FIG 6.5 NETWORK TOPOLOGIES



RINGS TRANSMITTING IN THE SAME DIRECTION



RINGS TRANSMITTING IN OPPOSITE DIRECTIONS

FIG 6.6 RING RECONFIGURATION

don't clash with any other messages. It is best used in systems where message density is low. The other method is to poll each station in turn until one wishes to talk and then hand over control. Reliability of this layout is higher than the star configuration and can be further improved by providing a redundant bus.

The ring configuration is similar to the multidrop line but the bus is unidirectional. It can be likened to a train where there is either a carriage available for each control station to put messages in or each station adds its own carriage if it has something to send. Because it is unidirectional its reliability tends to be lower than for the multidrop line, and improvement requires the addition of more hardware than just a redundant bus. Figure 6.6 shows the possible reconfiguration for two methods of improving reliability. One is for the use of both loops transmitting in the same direction and the second for each loop in different directions. The intelligence required in the interface to achieve this reconfiguring makes the ring more costly than the multidrop line.

6.3 System Design

Having examined communication technology as far as it affects a process control system we must now select the most appropriate system for our needs.

In Chapter two it was determined that the star configuration was too expensive and we have shown in section 6.2 that for comparable cost, the ring lacks reliability. Therefore the topology that is currently most suitable is the multidrop line. We must now choose the transmission medium, twisted pair or coaxial cable, and the arbitration system, contention or polling. These decisions are based on message density. We must also decide on a protocol for the system.

The messages and their estimated lengths in bytes and transmission frequencies per loop per second are shown in Table 6.1. To avoid burst mode noise as much as possible, messages should be short. The longest message (1000 bytes) should, therefore, be split up into packets. So allowing an information length of say 200 bytes and a protocol overhead of 10 bytes, each loop is transmitting 3.1 messages or 71.2 bytes/second. Allowing for a 10 byte response (acknowledgement) to each of these and a system with 100 loops, to effect transmission in half the available time requires a line speed of the order of 200 k baud. This is right at the upper limit of a twisted pair and therefore the allowed transmission distance (section 6.2) would be too short. For the 4 km that would be typical in a process plant and this speed requirement, coaxial cable is the best choice. As optic fibre technology improves, it would be expected to become the most common medium.

The protocol (link control or level two in our analysis) should, if possible, be effected in hardware, to increase overall system performance. It must also be as efficient as possible (for an overhead). This tends to favour the bit oriented protocol (6-6).

The system, therefore, uses a hardware implemented, bit oriented link control, transmitting over coaxial cable at a speed in excess of .2M baud. The network layout is a multidrop line and because of message density, using a polling arbitration method for highway control.

Table 6.1: System Messages

Message	Length	Frequency
Transmit a single data item	10 bytes	3 per second
Transmit a table of data for the operator interface	100 bytes	1 per 10 seconds (.1 per second)
Transmit a whole file to or from the engineer's interface	500 bytes	1 per hour ($3*10^{-4}$ per second)

REFERENCES

6-1 "Revised Data-Interface Standards"

D. Morris, Electronic Design 18, Sept. 1, 1977.

6-2 "Data Goes Faster, Farther with Chips for Drivers, Receivers"

D.A. Laws, R.J. Levy, Electronics, Sept. 14, 1978.

6-3 "LSI Ready to Make a Mark on Packet-Switching Networks"

G.L. Leger, Electronics, Dec. 20, 1979.

6-4 Telecommunications and the Computer

J. Martin, Prentice Hall, 1969.

6-5 Distributed Micro/Minicomputer Systems

C. Weitzman, Prentice Hall, 1980.

6-6 "Orient your Data-Link Protocol toward Bits, Though Characters Still Count"

A.J. Weissberger, Electronic Design 15, July 19, 1979.

6-7 "Errors and Error Control"

H.O. Burton, D.D. Sullivan, Proceedings of the IEEE, Nov. 1972.

CHAPTER 7

SYSTEM DATABASE

Figures

7.1 Database Design

This chapter deals with the design of a database for a distributed control system, what information should be stored, how that information should be stored and the types of interface that will be required for the engineer, the operator and the process.

First, however, the basic requirement of a database in general must be considered and then for a distributed database in particular. The operational requirements of a database are:

i) performance

ii) integrity

iii) security

iv) concurrency

Performance implies both efficiency, in time and space, and reliability. The database (DB) should respond to requests rapidly, whether these requests are for data modification or interrogation. The organisation of the data and the design of interface software are the two factors that most affect efficiency. The software overhead will slow the speed of response, but it has the advantage of providing greater data independence, and the data organisation can allow for rapid access to selected data without the need for traversing long paths through the DB.

Reliability includes hardware and software reliability. Hardware reliability has already been discussed in Chapter 2. Software reliability can be enhanced by the inclusion of effective recovery procedures and the ability to isolate those parts of the DB where an error has been detected, to prevent compounding of its effect.

The integrity of a DB is the safeguarding of the data in the DB. This is the purpose of the recovery procedures mentioned above. Standard approaches to preserving DB integrity are:

i) Journalling

ii) Checkpoint

iii) Backtracking

iv) Access control

v) Monitoring

Journalling is keeping a record of DB modifications; a checkpoint involves saving part of the DB so on error the DB can be restored prior to failure; backtracking involves reversing the effects of a transaction; access control is restricting access to user needs and monitoring is just checking for integrity breaches.

Security of a DB is a form of access control. Prevention of unauthorised access is usually by some sort of user codeword.

Concurrent access must be handled carefully to prevent deadlock or the use of erroneous data. Deadlock occurs when one user attempts to access a part of the database that is controlled by another user. Erroneous data will be produced if two users modify the same record simultaneously. Both of these situations are handled by a program called a locking monitor, that will make parts of the database exclusively available to one user and will contain checks to avoid deadlock.

Apart from operational requirements, a DB has design requirements:

i) data independence

ii) storage organisation

Data independence is desirable because it enables each user to see the DB from his own operational viewpoint. For example, the three interfaces in the control system each have a different way of presenting the data to the respective user, the process, the engineer or the operator. The more independence the data has, the less effect physical storage details have on interface software and so storage modifications require only localised

software changes.

There are three storage organisations or data models:

i) Relational

ii) Network

iii) Hierarchical

Relational data models can be imagined as multidimensional arrays of data. They contain no pointers so relationships between data are not determined by traversing the DB. Instead the relationship will be represented by a created array of the relevant information. A relational data model is simple for the casual user and so is very suited to management information systems. Its high operational software overhead and the fact that data in control system DB is typically not requested in terms of relationships means this data model is not very useful here.

The network data model has data stored in pockets in the DB. Associations between data are represented by pointers, and access to and through the DB is by means of these pointer paths. If the pointers are considered as mappings then a network data model is a many to many mapping.

The hierarchical data model is really just a special case of the network model. The corresponding mapping would be a one to many mapping. Compared to the network model, it has the advantages of simplicity of design and ease of use for limited cases but complex relationships require much traversing of the DB because access is limited to a top down search procedure only.

The last requirement for a general data base is information representation. An Infotech Report (7-1) states that real world relationships should be reflected in the data base to ensure flexibility of the system. If this were all that was done the result would be too complex to be

practical so the data is first normalized (7-2). Normalization is a reduction of complex real world structures to a simple and more regular form. Having done this it is then possible to define the data base on its two levels, the logical level or external schema and the physical level or internal schema. The external schema is the view of data base that the user has. As previously stated, this may be different for different users. The internal schema is dependent on the storage hardware available and the primary method of access (as this influences the choice of data model).

Having briefly described the considerations of a general data base we must now look at the extra constraints experienced with a distributed DB. Distribution affects the control of concurrent access and the mechanisms to provide security and integrity. It also adds to the problem of the location of data in the system. The simple solution is to place it as near as possible to the source and user of the data. This complicates the data base file directory design. The several possibilities are:

i) Localized file directory

ii) Distributed file directory

iii) Centralised file directory

iv) Extended centralised file directory

or combination thereof (7-3).

The last aspect of data base design that should be covered is the data language. This has two parts, the data design language (DDL) and the data manipulation language (DML). The DDL typically has two parts itself, one for the logical data definition and the other for the physical or storage definition of the data base. It enables the data base design to create the files and records that constitute the data base and in some cases also specify the links or relationships between these. Because of the fixed nature of the process control system, the DDL is of

interest, however the DML, as mentioned before, is the engineer's interface. The DML contains commands that correspond to operations that may be performed on the data base. The three types of command are:

i) Control - open, hold and release parts of the DB

ii) Retrieval - locate and/or gain access to files or records

iii) Modification - insert, delete, or update files.

7.2 Information Analysis

Before we can specify the most appropriate format for the control system data base we must look at the information that must be stored and where it should best be stored.

There are two distinct sections to the data base:

i) that part relevant to the process interface

ii) that part relevant to the operator's interface

The process interface contains files that direct the processing of each control station input. A file will have a header record specific to that input and several records, one for each routine used in the processing.

The header record should contain the following information:

i) file identifier

ii) input description

iii) input address

iv) status data

v) input units

The file identifier is necessary for the system. As a file within a data base it needs some label for addressing for access or modification. This can also double as a means of identification for experienced operators. To fulfil both functions effectively, it should be as short as possible to

reduce storage and communication overhead and yet still have some meaning for the operator. Conventional process control labelling goes part way toward achieving this, its major disadvantage being the limitation to six alphanumeric characters. The code typically uses three letters and three digits. The first letter identifies the type of measurement, T for temperature, P for pressure etc. The other two letters usually show the nature of the analog equipment used but here they no longer need serve that function. A typical identifier then would be TEC106, where T meant temperature and EC may mean ethanol column.

The description is needed for the inexperienced operator. As a supplement to the identifier it provides a fuller explanation of the nature of the input.

The input address is just the input channel number, or, for memory mapped I/O systems, the memory address.

Status data covers input on or off scan, the controller on local, manual or cascade and the output active or inactive. Not every input will need all of these but they cover the range of possibilities.

Input units are in engineering terms, degrees Celsius, PSIG or similar.

That is the header record; the routine records have:

i) routine identifier

ii) routine parameters

iii) output addresses

The routine identifier will be the name or number of the processing routine to be used. The parameters are the values needed in the routine equations and the output addresses show where the result should be sent. For an entirely flexible system where any parameter may be the output of any

routine, the address needs to have the file identifier, the routine identifier and the parameter number. This is why it is important to keep the file identifier as short as possible.

The other section of the data base is the operator interface files. These are:

i) the display files

ii) the log files

The display files, one per subgroup, consist of an identifier, a description and a list of input file identifiers that indicate the inputs in that subgroup. The log files have an identifier, a list of process variables to be included in the log and a print format. If desired, and this also applies to the process interface files, a scan interval can be included so that less important logs or inputs receive less attention.

7.3 Conclusions

The information required for the control system has been decided. Considering the data models discussed in section 7.1 one must now conclude the most efficient storage method and also, if it is necessary, where it should be stored.

The choice of storage method is dependent upon the situation in which it is used. Does the data need to be rapidly assessed? Does it need to be formatted for presentation? In what sequence is the information typically used?

Considering first the process interface, the data is required quickly and because of this it should not need formatting. The emphasis on speed precludes the possibility of data independence here, to reduce the software overhead. It also suggests that linked lists, for a network model,

would be undesirable. The preferred result, then, is to store the file in physically sequential RAM. There is no overhead for pointers, the data is easy to access directly and the file is usually worked through from top to bottom in use. A local directory would be used to show the files in each control station. The result is shown in figure 7.1a.

The operator interface has a less stringent speed restriction and therefore can be more independent, but it is still typically accessed sequentially. The hierarchical data model is still the preferred choice (Fig. 7.1b).

It should be noted that all of the input file is not required at the process interface. Some is only relevant to the operator and therefore the description and input units may be stored at the operator interface, saving control station storage.

Extra information required in a computer control system is:

i) hardware status information

ii) historical input data for drawing input trend graphs

Both of these are used, and so should be stored, in the operator interface. The historical data, because of bulk, must be stored on a backup storage medium such as a floppy disc rather than in RAM.

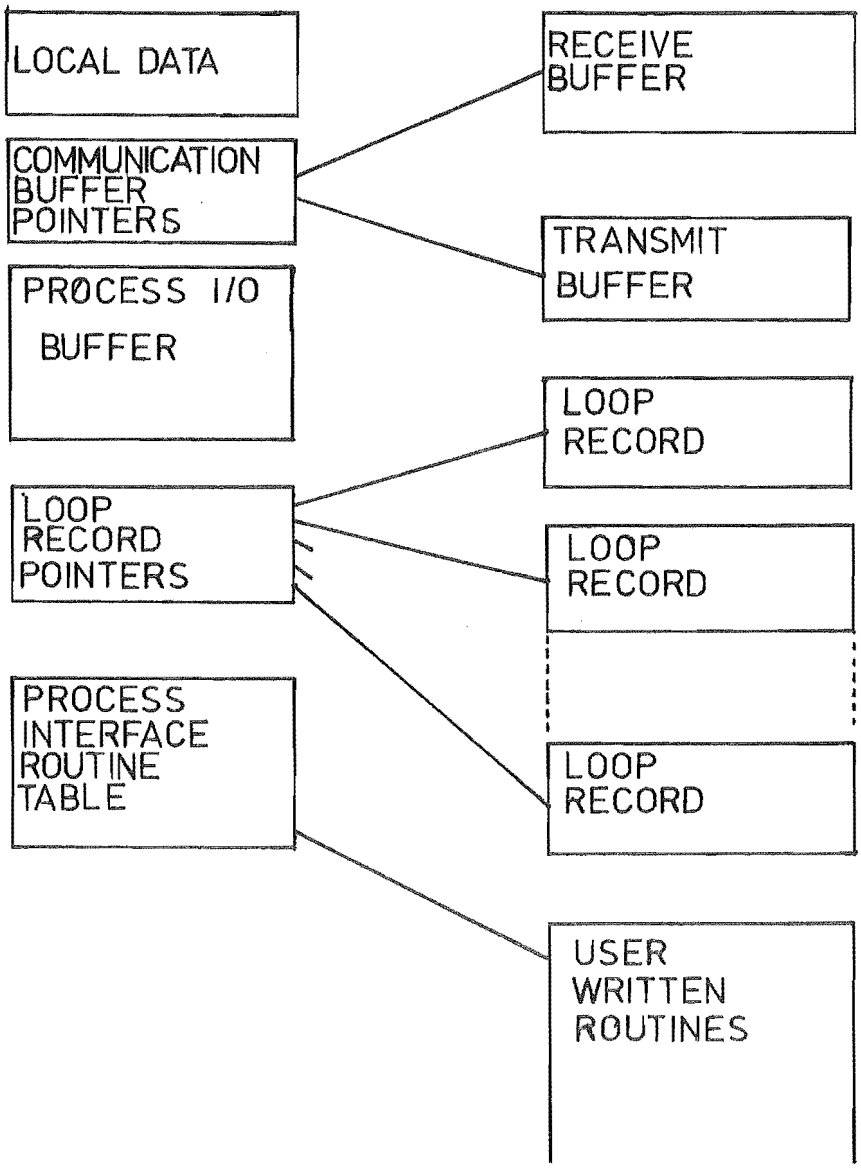


FIG 7.1A CONTROL STATION DATABASE

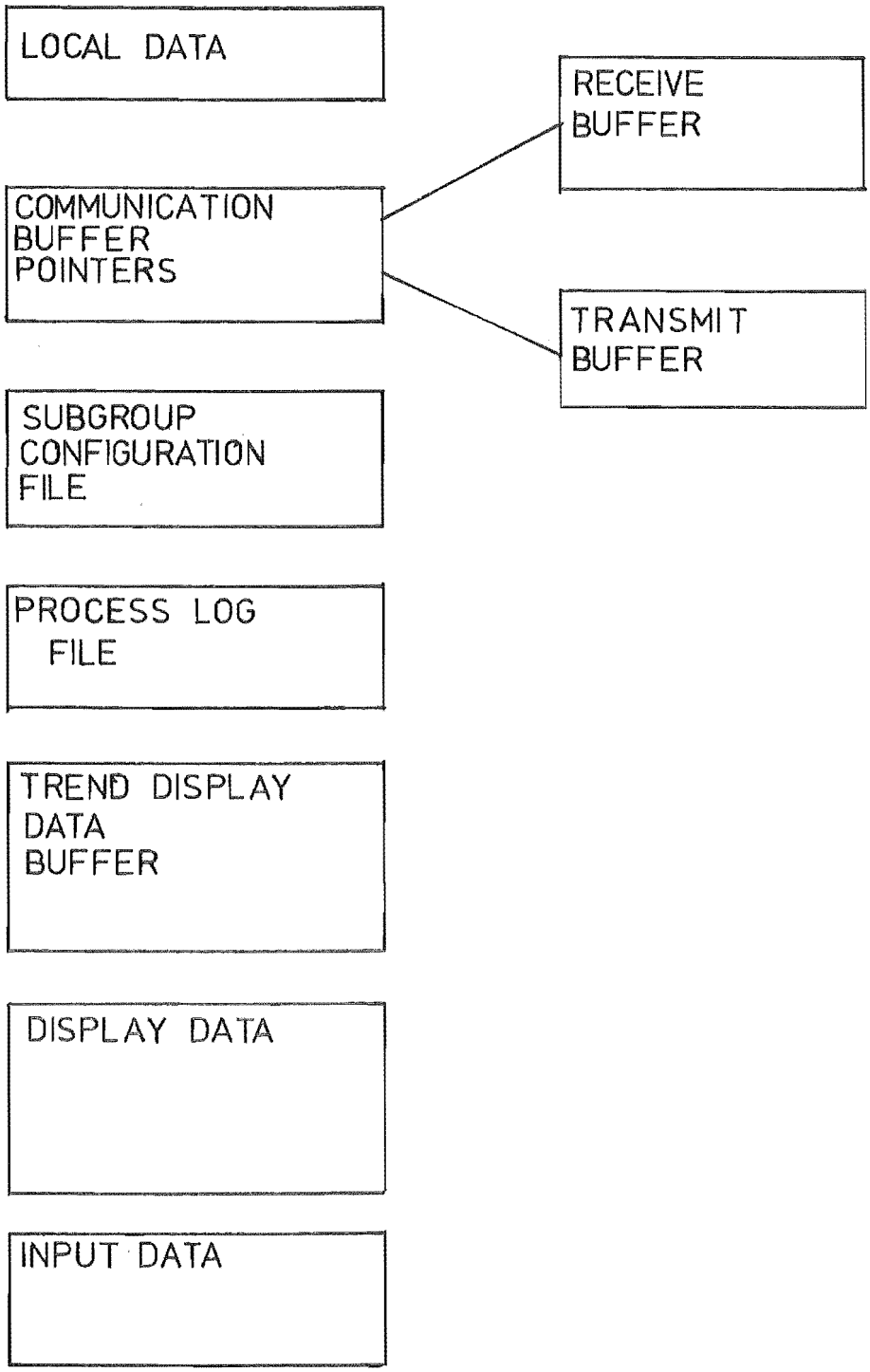


FIG 7.1B OPERATOR INTERFACE DATABASE

REFERENCES

7-1 Data Base Systems

Infotech State of the Art Report, 1975.

7-2 Data Base Management Systems

D.C. Taichrityis, F.H. Lochovsky, Academic Press, 1977.

7-3 Distributed Micro/Minicomputer Systems

C. Weitzman, Prentice Hall, 1980.

CHAPTER 8

IMPLEMENTATION

Figures

8.1 Introduction

The previous seven chapters have provided a basic specification which can be used to design a distributed control system. This was done in the Chemical Engineering Department. Economic constraints prevented a faithful implementation, and deviations will be pointed out in the relevant sections.

One control station, based on Motorola's 6800 μP, was used. The two man/machine interfaces, the engineer's interface and the operator's interface were based in the department's PDP11 minicomputer. The communications line was a 20 mA current loop operated at 2400 baud.

8.2 Control Station

Hardware:

The control station was based on an M6800 evaluation kit that had already been used within the department. Table 8.1 gives a list of hardware components in the system (8-1).

The 8 bit CPU was a little slow. When the system was run with control interrupts every half second, the communications system performance was degraded to the point where some messages were missed. Had the dual CPU layout, as suggested in Chapter 3, been used, this problem would have been avoided. Because of this time limitation, there was no possibility of a suitable local operator interface being incorporated. The lack of the second CPU also decreased the reliability of the station (as per Chapter 3).

The use of the APU enabled the system to perform floating point arithmetic and compute transcendental functions (see Table 8.2). These increased the possible range of signal processing routines that were available to the engineer. The unit's primary advantage however was in allowing the use of full floating point rather than scaled integer arithmetic.

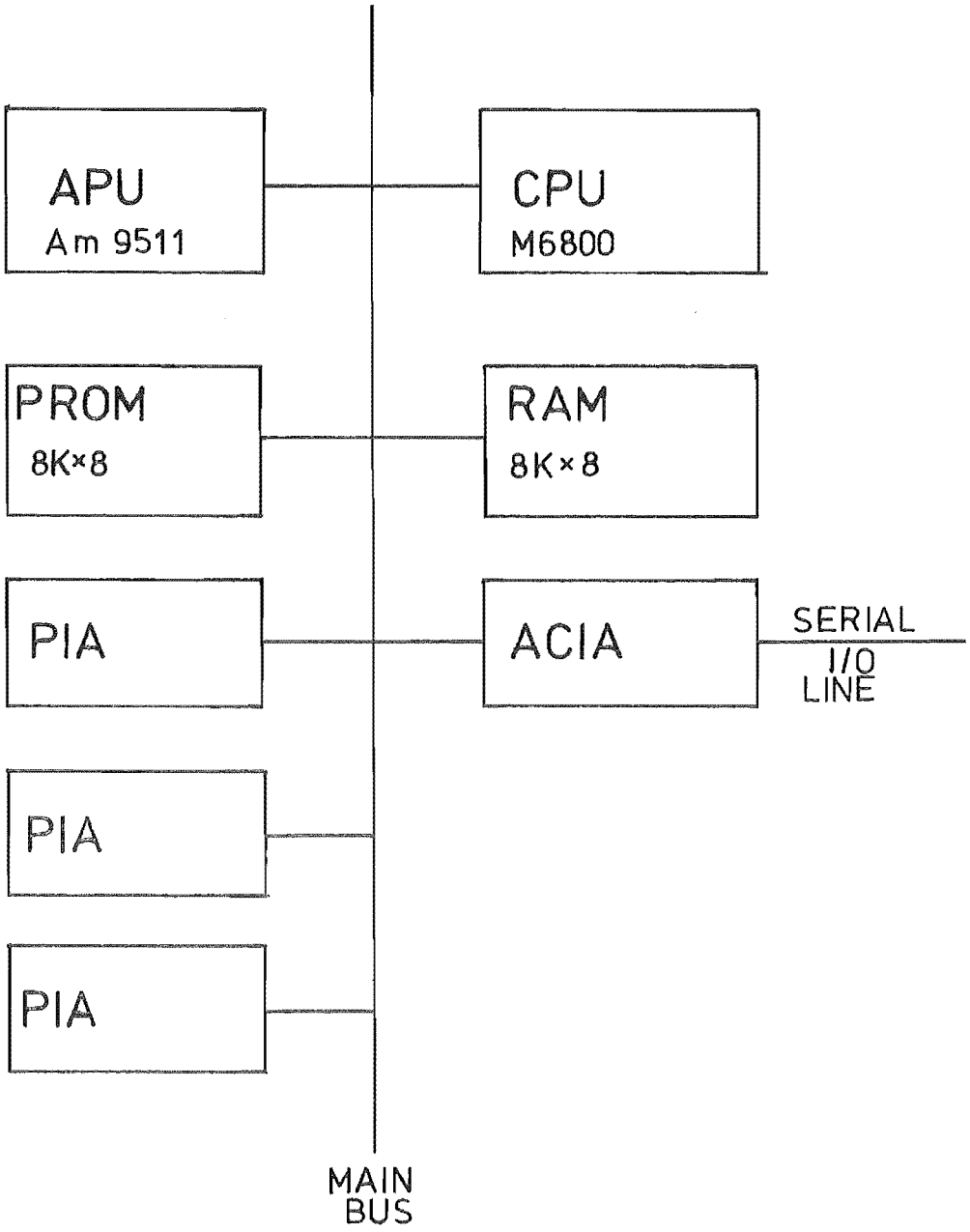


FIG 8.1 MICROCOMPUTER HARDWARE

Table 8.1: M6800 Microcomputer

	M6800	Control processor unit (CPU) 1 MHz 8 bit
	Am9511	Arithmetic processor unit (APU)
	8K x 8	Random access read/write memory (RAM)
	8K x 8	Eraseable programmable read only memory (RDM)
	MC6850	Asynchronous communications interface adapter (ACIA)
3*	MC6820	Peripheral interface adapters (PIA's)
	SDM853	16 channel 12 bit data acquisition module (DAM) 33 kHz
4*	MC3408	8 bit D/A converters

Table 8.2: Am9511 Transcendental Functions

Square root	$\sqrt{x}$
Sin	Sin x
Cos	Cos x
Tan	Tan x
Inverse sin	Sin^{-1} x
Inverse cos	Cos^{-1} x
Inverse tan	Tan^{-1} x
Log	$\log_{10}$ x
Natural log	$\ln_e$ x
Exponential	e^x
Power	x^y

The Burr Brown data acquisition module was not the best choice of input system for a control station. It interfaced through one of the PIA's and used the available handshake lines for control (see Fig. 8.2). A preferred system would have been either a memory mapped input unit, where one input channel is represented by an address on the bus, or a unit interfaced through DMA (direct memory access), which would have effectively been memory mapped. However, at 33 kHz throughput rate, it was fast enough to present no problems for the CPU. The 12 bit resolution, giving 1 part in 4096 or .025%, was more than adequate for the transducers to which it was connected (8-1).

The D/A's used for output were interfaced through the remaining PIA's (see Fig. 8.2). Memory mapping would again have reduced the hardware count and thereby increased reliability but the method used presented no problems.

The process inputs and outputs were all standardized to 0-10V but one of the D/A's had an optional 10-50 mA output.

The communication system interfaced with the control station through the ACIA. The line speed could be software set, up to a maximum of 2400 baud. The ACIA provided only parity as means of error checking, and effected no other protocol function as described in Chapter 6.

The only other hardware functions were; the provision of 5 light emitting diodes (LED's) on the front of the microcomputer, which were used to indicate which section of the software was active, a set of LED's on the data acquisition module, which, because of the speed of operation, only served to indicate that the unit was operational, and an analog panel meter, switch selectable, to view the D/A output signals. This human interface was only useful as a diagnostic tool but as mentioned earlier, the CPU was too slow to incorporate a more comprehensive operator interface.

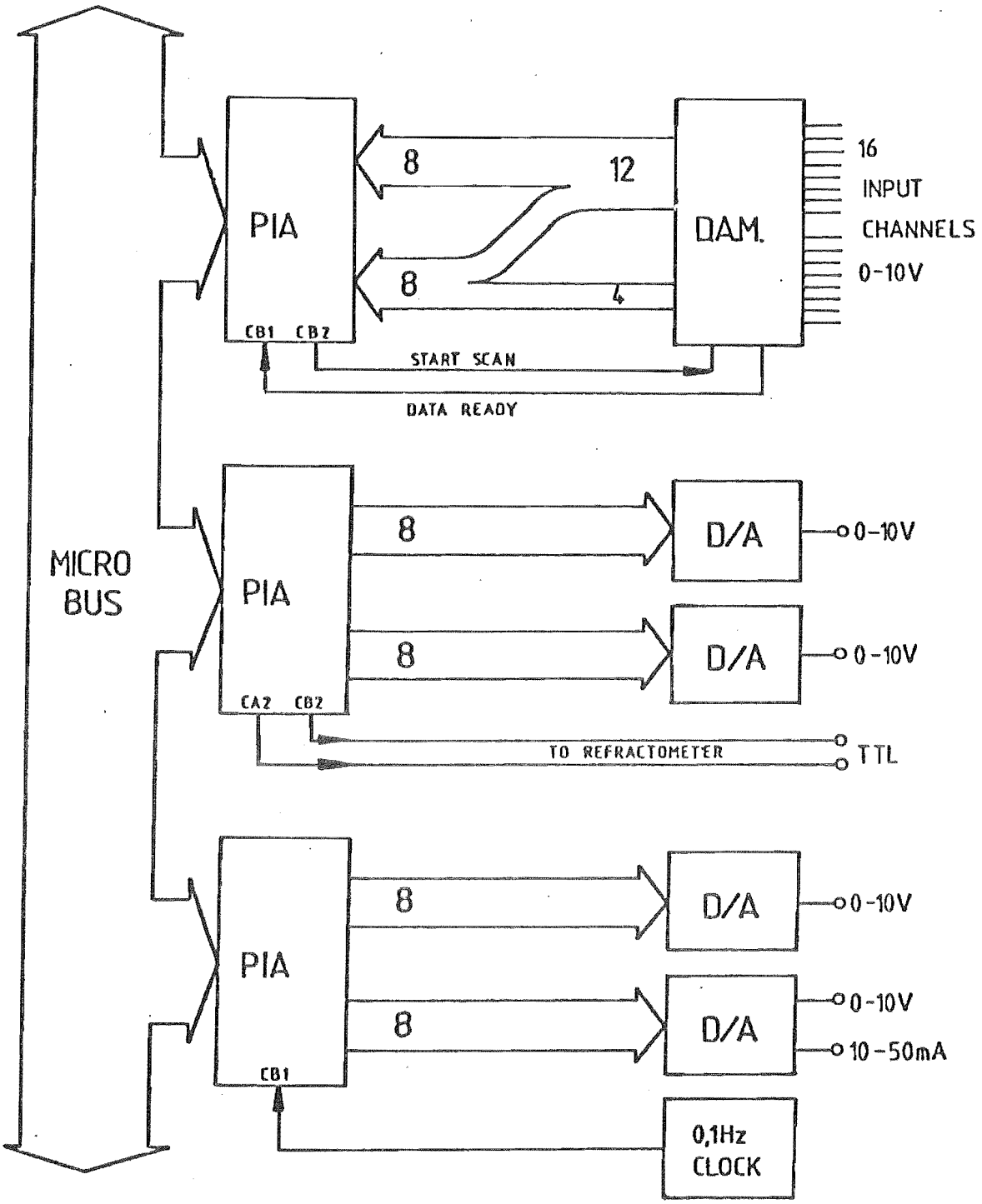


FIG 8.2 COMPUTER PROCESS INTERFACE

Software:

The software for the process interface was locally developed using the Department's PDP11 minicomputer (8-1). Once assembled, the routines were burned into AMI S6834 512 x 8 EPROM's using facilities within the evaluation kit. The software could be divided into three distinct parts:

i) System initialization

ii) Communications

iii) Process interface

A complete listing is given in Appendix C.

The system initialization routine

i) set up the three interrupt vectors

RST - reset

NMI - non maskable interrupt

IRQ - interrupt request;

ii) configured the ACIA and the three PIA's;

iii) zeroed RAM;

and iv) initialized the process interface database.

The RST and NMI interrupt vectors caused a return to the system initialization routine and an IRQ interrupt activated a polling routine, which checked to see if the interrupt was from the control system or the communications system. If it was neither, it assumed a reset and returned to the system configuration. There proved to be problems with this layout. The communications system would occasionally cause the microprocessor to temporarily halt. To correct this it was necessary to RST and resend the control system database. The problem was found to be due to interactions between the communications and process subsystem interrupts, and would not have occurred with a dual CPU control station, or a hardware vectored interrupt system (which the M6800 evaluation kit did not have). To prevent the communication system from

interferring with the control system, once a timer interrupt (generated every half second) had been received, the interrupt polling routine sent a busy message to the PDP11. This would have been totally unsatisfactory had there been more than one control station on the communication line, and unnecessary with a dual CPU system.

The communications software will be discussed in the next section.

The process interface and control system software were activated every half second, by an externally generated timer interrupt. The software consisted of:

i) a timer routine to handle clock interrupts (see Fig. 8.3 for flow chart);
ii) an input routine to begin loop processing (Fig. 8.4);
iii) the process interface routines (listed in Table 8.3);
iv) an output routine (Fig. 8.5).

The database consisted of timers, pointers, counters and temporary storage, communications buffers, loop record storage and routine storage (primarily the ring buffers for the delay routines). The complete loop record is listed in Table 8.4, and the database layout is shown graphically in Fig. 7.1 and listed in Appendix C. A loop record occupies 634 bytes of storage (so the eight loops for one control station need 5 k bytes) and of this the routine output addresses are 45%. The desired flexibility of the control system, as discussed in Chapter 4, necessitates that the output of any routine can be used as the input to any parameter within any routine. Because of this, the output address must make provision for the record identification, the routine number and the parameter number within that routine. The storage requirement for a loop record could be reduced markedly if the record identification was restricted to three characters (at present there is an allowance of six) or could be expressed numerically.

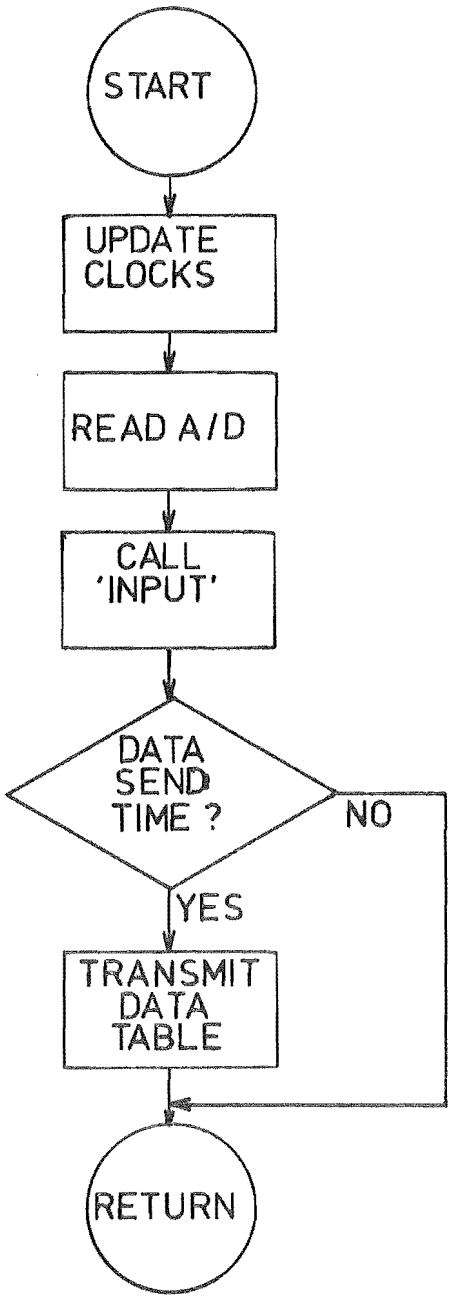


FIG 8.3 TIMER ROUTINE

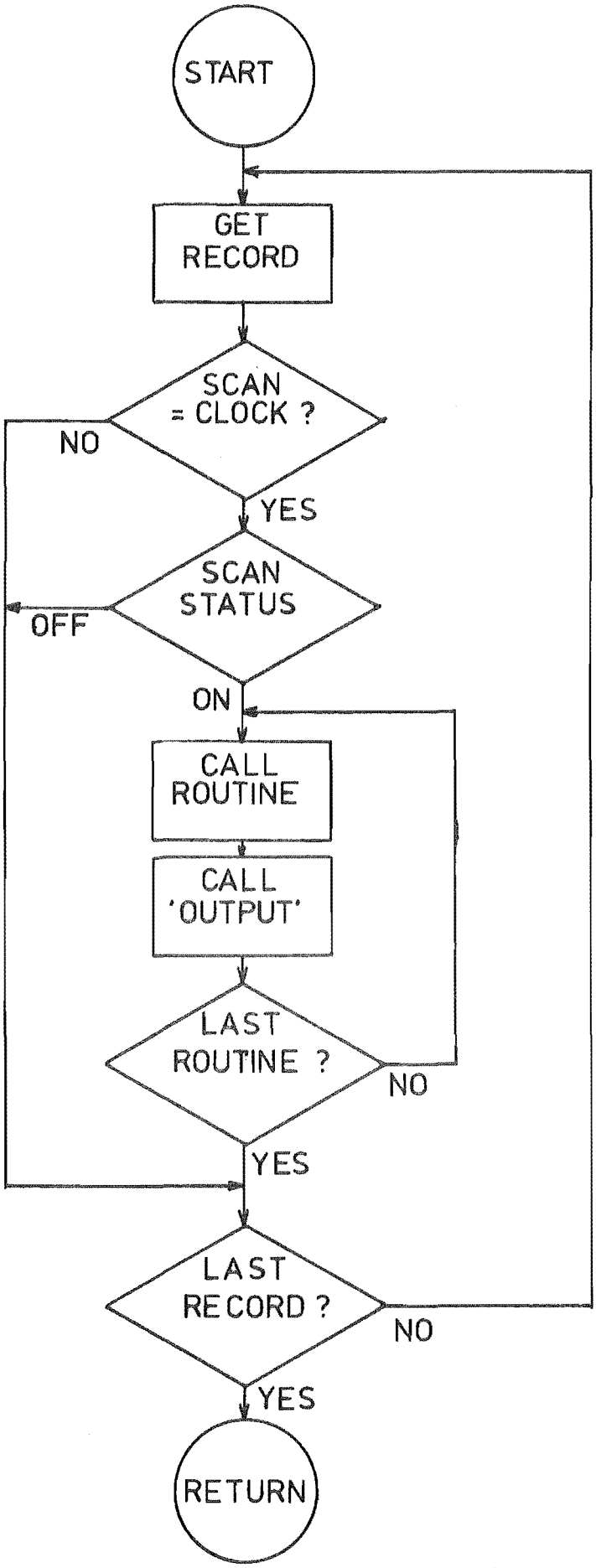


FIG 8.4 INPUT ROUTINE

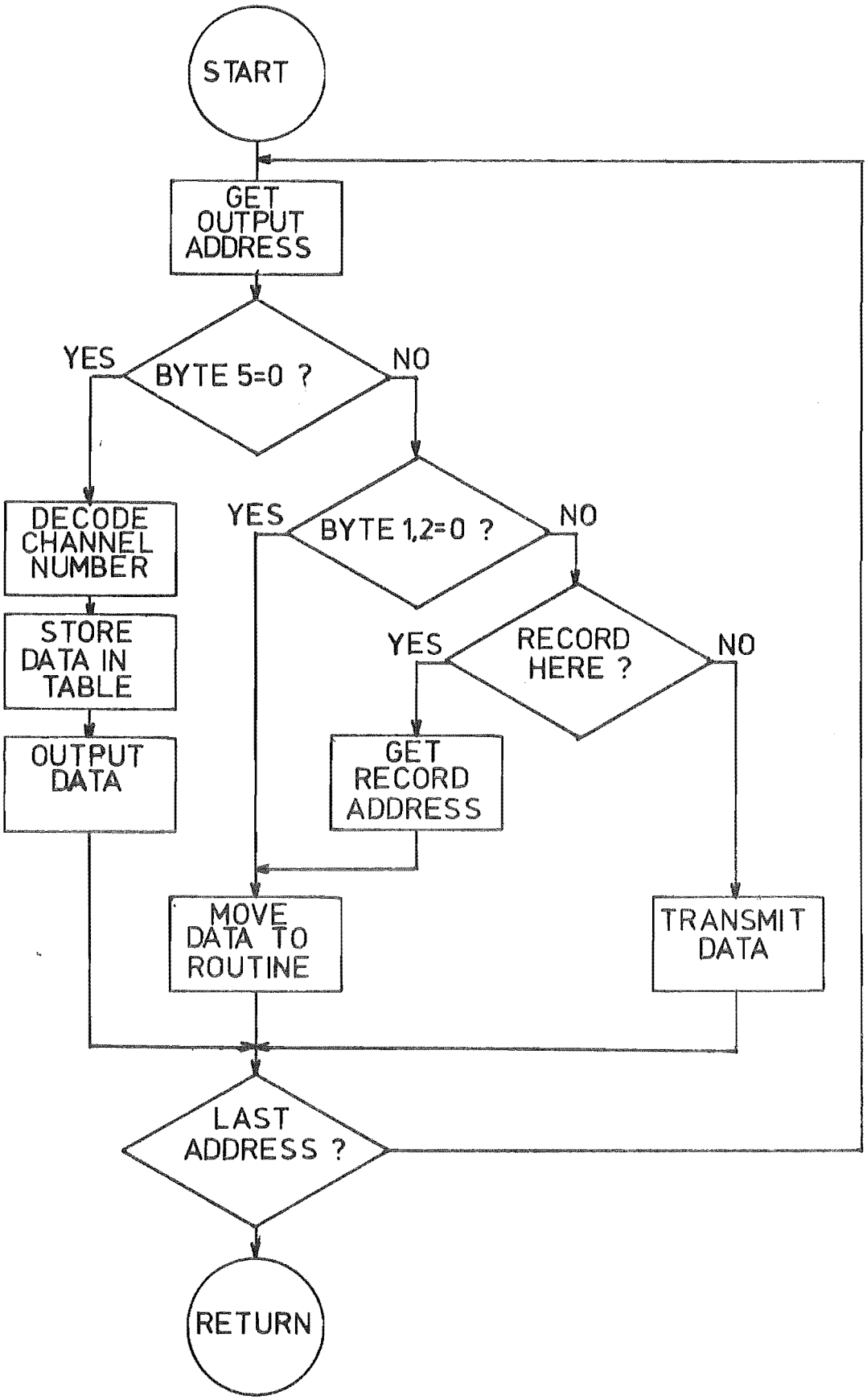


FIG 8.5 OUTPUT ROUTINE

Table 8.3: Process Interface Routines

Exponential filter	$y_i = y_{i-1} + \alpha(x - y_{i-1})$
Linear scale	$y = Ax + B$
Normal log	$y = \ln x$
Exponential	$y = e^x$
Sin, Cos, Tan	$y = \text{Sin}(x), \text{Cos}(x), \text{Tan}(x)$
Inverse Sin, Cos, Tan	$y = \text{Sin}^{-1}(x), \text{Cos}^{-1}(x), \text{Tan}^{-1}(x)$
Inverse Sinh	$y = \text{Sinh}^{-1}(x)$
Square root	$y = \sqrt{x}$
Quadratic	$y = Ax^2 + Bx + C$
Power	$y = Ax^B$
Add-Subtract	$y = Ax_1 + Bx_2$
Multiply-Divide	$y = Ax_1x_2/x_3$
Integral	$y = \int x\, dt$
Derivative	$y = \frac{dx}{dt}$
Alarm, absolute and deviation from set point	
PID on error PID on process variable	) velocity for
Minimum	$y = \min(x_1, x_2, x_3, x_4)$
Maximum	$y = \max(x_1, x_2, x_3, x_4)$
Deadband	$y = x$ if $x < LO$ or $x > HI$ $= 0$ if $LO < x \leq HI$
Limit	$y = HI$ if $x > HI$ $= x$ if $LO < x \leq HI$ $= LO$ if $x < LO$

Table 8.3 contd.

Comparator		$y = A$ if $x > C$
		$= B$ if $x < C$
Output Limit		$y_i = HI$ if $x > HI$
		$= x$ if $LO < x < HI$
		$= LO$ if $x < LO$
	and	$\Delta y_i = \Delta max$ if $\lvert\Delta y\rvert > \Delta max$
		where $\Delta y_i = x - y_{i-1}$
Set point ramp	if	$\Delta y_i = SP - x > \Delta max$
(for cascade controllers)		$\Delta y_i = \Delta max$
		$y_i = y_{i-1} + \Delta y_i$
Delay		$y_i = x_{i-n}$ where $n < 63 * \tau$

KEY

y_i = output

x = input

HI, LO, A, B, C = constants

τ = sample interval

Table 8.4: Loop Record

Header:	identification		4 bytes
	description		32 bytes
	input and output units		20 bytes
	scan interval		1 byte
	data input address (parameter number)		1 byte
	input channel number		1 byte
	loop status		1 byte
	raw data		4 bytes
			64 bytes
For each interface routine (up to 12)			
	routine number		1 byte
	number of outputs (up to four)		1 byte
	output address (6 bytes each)		24 bytes
	result		4 bytes
	routine parameters (up to four)		16 bytes
			46 bytes/routine

Allowing for extra data storage, there is a total of 634 bytes/loop.

Each output address has

Record ID		4 bytes
Routine number		1 byte
Parameter number		1 byte

The status byte has

On/off scan	bit 0	1 = On
Auto/manual control	1	1 = Auto
Cascade/local control	2	1 = Cascade
Cascade flag	3	1 = Not cascade

This would cut the storage for the identification from 4 to 2 bytes and the total record storage to 536 bytes (a 15% saving). With the present cost of RAM, however, the restriction to achieve this saving does not seem warranted. It would also be possible to save a small amount of memory by keeping the description and input and output units in the operator interface as they are only used there. The extra processing required in the engineer's interface to then examine a complete record suggests, once again, that the small saving is not worthwhile. Also, with RAM soon to be available in an 8K x 8 chip (it is already available in the USA) the attempt to save a small amount of memory is unnecessary.

The next point about the loop record that should be discussed is that it is fixed length. There are 12 routines, each of which can have four output addresses and four parameters. It would seem better to use a linked list format and remove these restrictions. To allow this it would be necessary to have a routine that collected up the free pieces of storage after a loop record was detected, so as to make maximum room for a new record. To shift three records, a worst case, a routine capable of this would take approximately .2 seconds (assuming a routine of 20 instructions, using 2 instruction cycles each, shifting 2000 bytes). This is too long compared to the .5 second process control cycle time. A faster CPU (e.g. the 6809 which operates at 5MHz) or a dual CPU layout would enable a linked list storage structure.

The Timer routine increments all clocks, one for each loop and one for the data table. It then activates the DAM and for each channel, checks the channel address and if correct moves the data to the data input buffer. If the channel address is wrong an error flag is raised and three errors causes a hardware fault message to be sent to the operator's interface.

Once the data from DAM has been entered into the buffer, the Input routine is called. The last thing Timer does is to send the input data table if the time is right.

The Input routine cycles once for every loop record. If the scan status bit is off it will skip that record. If the loop is 'On Scan', the Input routine next checks the time. If that loop's clock matches the record scan interval the record will be processed. For each process interface routine, the routine will be called. A computation error results in the previous result being used and an alarm message sent. The Output routine is then called. This cycle is repeated until the end of record marker is reached.

The Output routine responds to the coding in the output address. It is:-

Bytes 1-4 Record ID

if bytes 1, 2 = 0 this record

Byte 5 Routine number

6 Parameter number

if byte 5 = 0, the output belongs to the real world.

Byte 2 gives the output address channel number.

Output Channel numbers

0	Process variable
1-9	Trend recorders in the operator interface
10-14	D/A outputs

8.3 Communication System

Hardware:

The communication system made use of the microcomputer-minicomputer interface hardware that already existed (8-1). The line was a twisted pair and the transmission method, a 20 mA current loop. At the microcomputer end the line was connected to the ACIA and at the PDP end it interfaced to a DR11C parallel input via an MM5303 UART (8-1). The line speed was set at 2400 baud and could be changed by a switch at PDP end and by software or a switch in the microcomputer.

Software:

Because of the byte oriented hardware, bit oriented protocols such as HDLC were ruled out. An adaptation of the IBM Bi-sync protocol was used. The message format is shown in Figure 8.6. The header includes: a character count, a destination address, a source address, and a control byte. If the control byte was zero, the message was a system message rather than a data message. The system messages were: ACK, acknowledging correct receipt of a message, NAK, indicating the message contained an error, POLL, to allow the control station to start transmission, RESET, to cause the control station software to enter the system initialization routine, FINISHED, to indicate that a control station had finished transmitting its queued messages, and NOT READY or YES, to state whether or not the control station was ready to accept a message. The next part of the message was either the data or a control character, to indicate one of the above system messages. The last byte was an error check byte. Error checking was done in two ways. On each byte transmitted, the UART or the ACIA did a parity check and the final error check byte was the exclusive OR (XOR) of every byte in the message. This is not as efficient as a cyclic redundancy check (CRC, see section 6.2) but the software overhead needed to compute

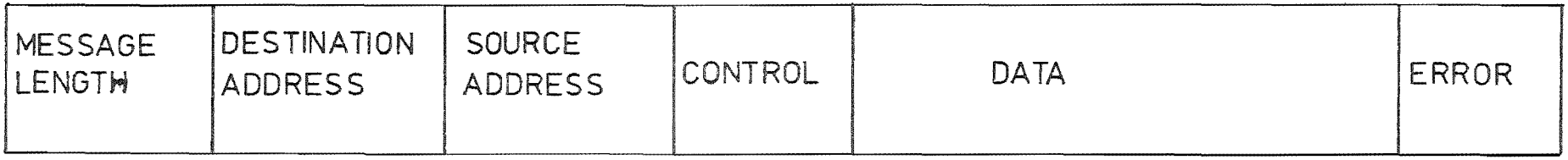


FIG 8.6 MESSAGE FORMAT

the CRC precluded its use in the microcomputer. This message format was found to be uncomplicated and effective for this situation. With the low line speed, however, any more control stations would have resulted in problems. The message density assumed in section 6.3 (Table 6.1), for the four control loops used in this system, used all the available time. With another control station, messages would have backed up or never been sent. A dual CPU control station, a higher line speed, and one of the protocol chips mentioned in Weitzman (6-5) would have improved the throughput of the system.

The software had three layers. The lowest level effected the transfers to and from the UART or ACIA, and did the error checking. The next level had two routines, one for receiving a message, and one for transmitting a message. The receiver was called up by an interrupt, generated by the first received character, the message length. The message length is then passed to the lowest level routine which reads in the rest of the message. The destination address is checked to see if the message was for this address. If not, no further processing is done. If the message was correctly addressed, the error flag is checked. An error in a non system message results in NAK being sent. For an error free non system message an ACK is transmitted to the message source. Finally the appropriate servicing routine is called.

The transmitter is activated on receipt of a polling message. The transmit queue is five messages long. For each message, the message is transmitted with an error check byte appended. A watchdog timer is then started. An ACK returned before the timer stops results in the next message in the queue being sent. A timeout or NAK will cause a retransmission of the same message for up to three attempts. If all three attempts are unsuccessful, a highway fault message is transmitted to the operator's interface.

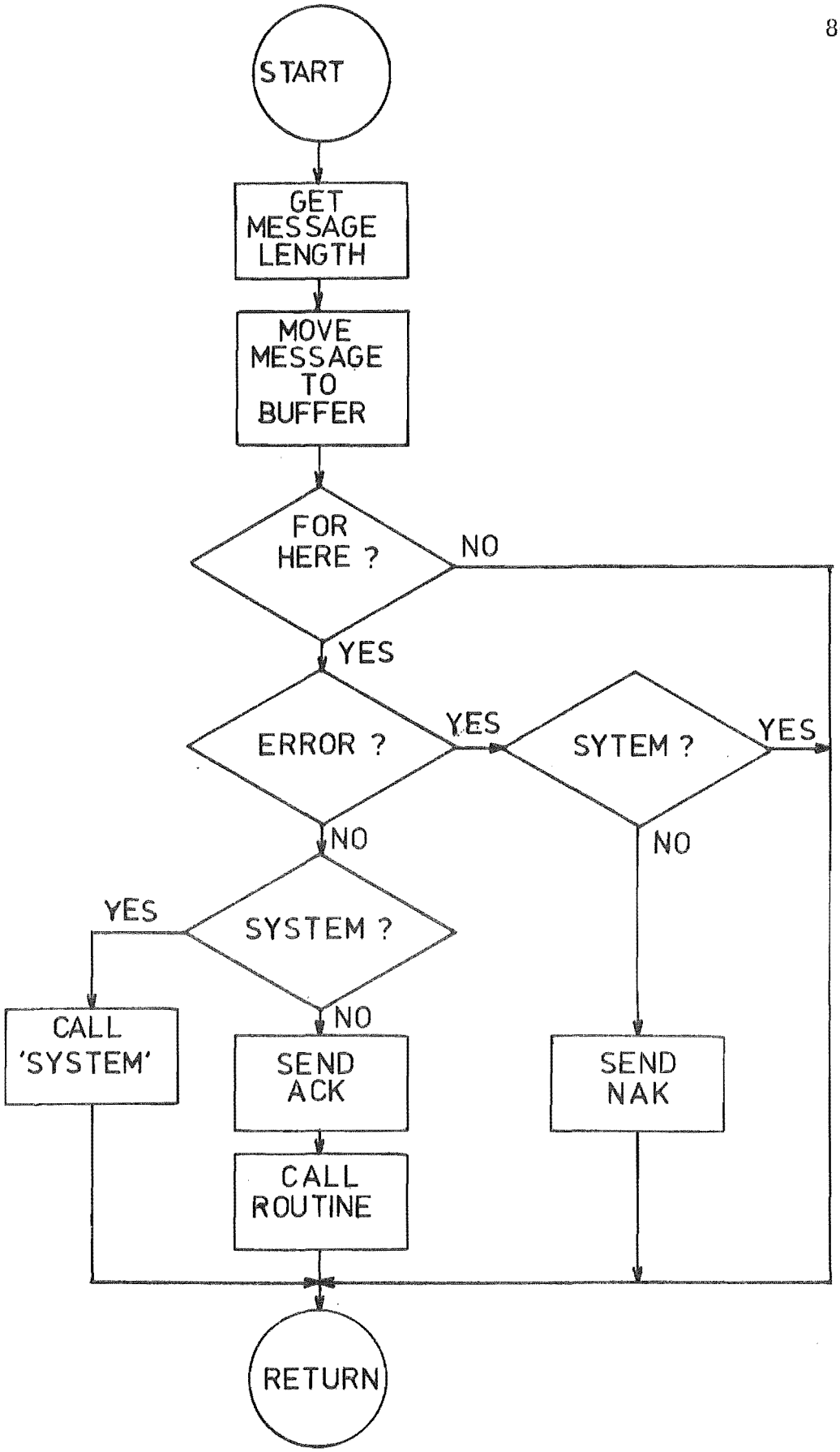


FIG 8.7 RECEIVE ROUTINE

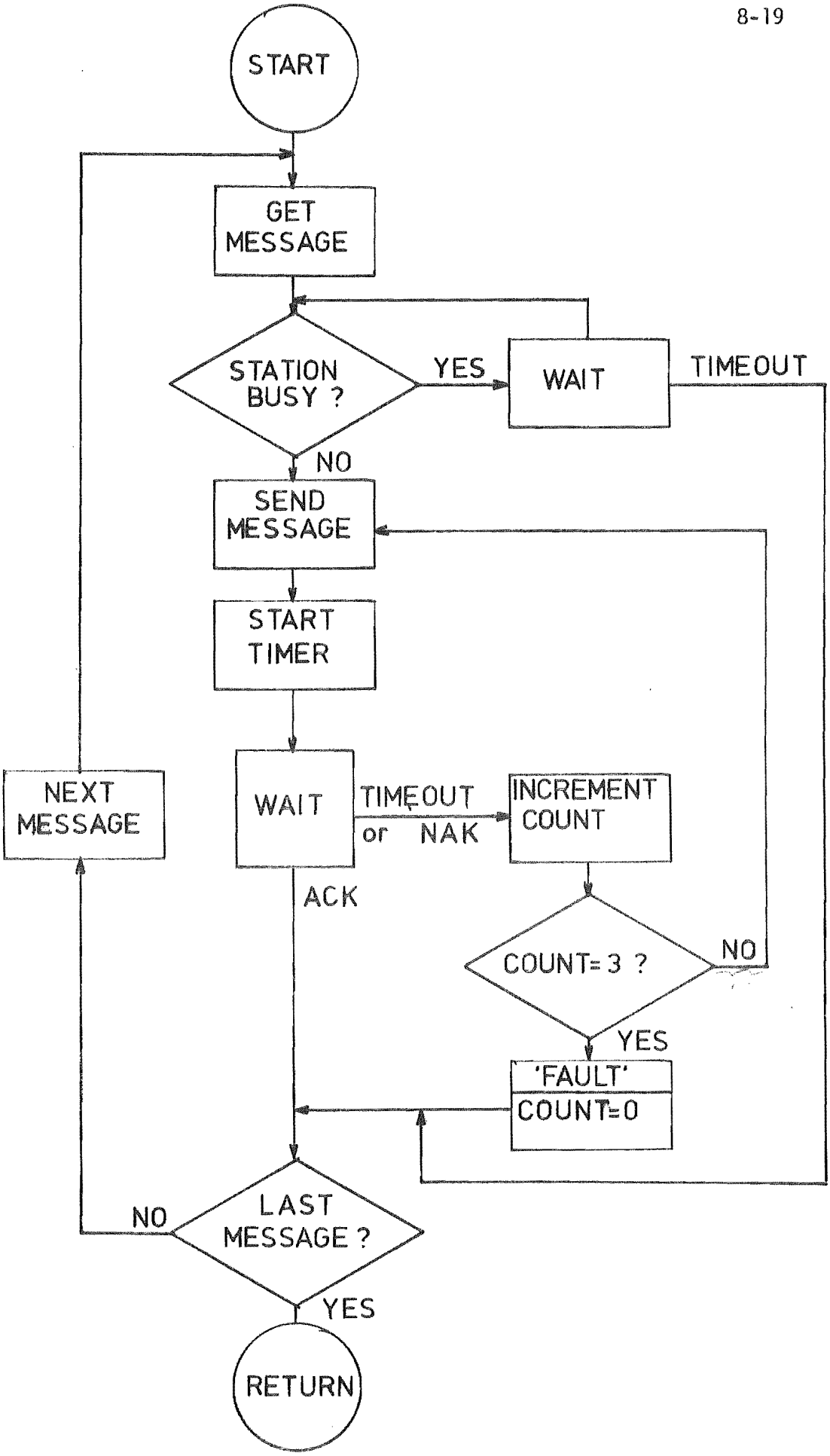


FIG 8.8 TRANSMIT ROUTINE

There are two groups of servicing routines, one for the control stations:

i) PUTDAT, for the storing of individual pieces of data;

ii) PUTREC, DELREC, GETREC, to enter into, delete from or fetch from the control database, a loop record;

iii) PUTRTN, to add user written routine to the process interface;

and one for the operator interface:

i) ALMNOT, to notify the operator of an alarm limit violation,

ii) HARDST, to act on a hardware or highway fault message;

iii) TRENDT, to store data about process variables periodically sent from the control stations;

iv) PUTfil, to copy records pertaining to subgroup configurations, control loops, and process logs into the interface database.

One other service routine common to all points on the highway processed system messages.

This three-layered approach to the communication software was found to be satisfactory. In particular, modifications could be made to the system without affecting all levels of the software. In fact, the author changed the message protocol, which only necessitated changes to the receive and transmit routines.

8.4 Engineer's Interface

Hardware:

The engineer's interface was based on the department's PDP11/15 minicomputer, with 28 k words of memory and a 2.5 M byte removeable hard disc (RK05 Discpack) backup store. Communication was through a VT100 visual display unit. A standard QUERTY keyboard was used for input.

Software:

To begin, a routine called MAIN was used to generate the database disc files for the control station and operator interface. It also sent a RESET message to the control station. The engineer's interface permitted the following functions:

i) add a new control station to the system;

ii) create a subgroup configuration record;

iii) create a process log record;

iv) create a process interface control loop record;

v) modify a database record;

vi) delete a database record.

The control loop records were explained in section 8.2. Table 8.5 shows the format of the subgroup configuration and process log records. The input of the necessary data was carried out in a conversational on-line manner. The initial display was:

```
SELECT DESIRED FUNCTION - <RETURN>
      VALID SYSTEM INPUTS ARE:-
      SUBSYSTEM - TO CREATE SUBSYSTEM DISPLAY RECORD
      LOG       - TO CREATE DATA LOGGING RECORD
      PROCESS   - TO CREATE PROCESS RECORD
      MICRO     - TO INPUT ADDRESS OF NEW CONTROL STATION
      ALTER     - TO MODIFY AN EXISTING DATABASE RECORD
      DELETE    - TO DELETE AN EXISTING RECORD
      ESCAPE    - TO END PROGRAM
SELECT DESIRED FUNCTION -
```

The desired function was then selected by typing in at least the first two letters of the chosen function.

The delete function was:

Table 8.5: Database Record Descriptions

Log Record

identification	4 bytes
print interval	2 bytes
print format	150 bytes
(150 characters)	
loop identifiers	40 bytes
(up to 10 loops)	

including spare bytes, a total of 200 bytes per record.

Subgroup Record

identification	4 bytes
description	32 bytes
loop identifiers	32 bytes
(up to 8 loops)	

including spare bytes, a total of 72 bytes per record.

INPUT ID CODE OF RECORD TO BE DELETED ABC123

After the code had been entered, the response was either RECORD NOT FOUND or a return to the initial display.

For a subsystem display (new screen):

INPUT SUBGROUP CODE (6 CHARACTERS) SUB1

SUBGROUP DESCRIPTION (32 CHARACTERS) SUBGROUP ONE

RECORD ID LIST (8 RECORDS, BLANK → LAST ONE)

ABC123

ANY MORE? Y or N

The initial display was returned to after the last record identification had been entered.

A data logging record (new screen):

INPUT LOG NAME (6 CHARACTERS) AND PRINT INTERVAL (HOURS)

ONE PER LINE

LOG 1

2

INPUT RECORD ID LIST (10 RECORDS, BLANK → LAST)

ABC123

INPUT PRINTING FORMAT, LIMIT OF 150 CHARACTERS

[= NEW LINE

]= LAST CHARACTER

= DATA, USING NEXT RECORD ID

[SHIFT OUTPUT = #]

ANY MORE? Y or N

The micro function:

INPUT HIWAY ADDRESS OF NEW CONTROL STATION 04

ANY MORE? Y or N

To create a process record:

INPUT ADDRESS OF CONTROL STATION FOR WHICH THIS RECORD IS INTENDED 04

For each part of the process record the screen was blanked. This enables the engineer to concentrate on one section at a time. The header:

INPUT THE FOLLOWING INFORMATION
RECORD ID 6 CHARACTERS ABC123
DESCRIPTION 32 CHARACTERS ETH COLUMN TOPS TEMP
PV UNITS 10 CHARACTERS DEG C
V UNITS 10 CHARACTERS %
SCAN INTERVAL 2
DATA PARAMETER NUMBER 2
INPUT ADDRESS (CHANNEL NUMBER) 1

The scan interval was in units of 5 seconds. The data parameter number referred the parameter in the first process interface routine for this record, that the data collected from the input channel, should be put into. For each routine the engineer was first asked:

SEE ERRATA

INPUT ROUTINE NAME (OR HELP OR END) FI

Here, as with the initial display, only the first two characters were required. A list of relevant data about that routine was then written out and the value of parameters was input:

FILTER $y = y(I-1) + A * (IN - y(I-1))$
VARIABLE RANGE
y = NEW FILTERED VALUE UNLIMITED
y(I-1) = OLD FILTERED VALUE UNLIMITED
IN = NEW INPUT UNLIMITED
A = FILTER CONSTANT 0 - 1

```
PARAMETER       OUTPUT ADDRESS
A               1
IN              2
INPUT  A        0.8
```

The first line gives a definition of the routine. Next follows an explanation of the variables, and their permitted ranges. The output address gives the data required for use in the output address, to enable any parameter to the output of any routine. After the initial value of each parameter:

```
NUMBER OF OUTPUTS (MAX 4)  2
OUTPUT CHANNEL NUMBER (-1 IF NO CHAN)  0
OUTPUT CHANNEL NUMBER (-1 IF NO CHAN)  -1
RECORD ID (<RET> IF THIS RECORD)  <RETURN>
ROUTINE NUMBER  2
PARAMETER NUMBER  3
```

The routine number referred to the routine in the record. In this case, the output went to the second routine in this record, the third parameter. A full list of records, with their explanations is given in Appendix D, and was written out if HELP was typed in. For a definition of the output channels, see the end of section 8.2.

The modification acted similarly, no matter what type of record was to be altered:

```
INPUT ID CODE OF RECORD TO BE MODIFIED  ABC123
```

The record was then written out, one line at a time.

```
RECORD ID       ABC123   /
```

The response could be either

```
<RETURN>  -  leave as is,
 /        -  delete data, or
```

ABC124 - replace present data with this new data.

This was done for all the information in the record. With the routines in the process record, however, more information was provided for the engineer.

ROUTINE DATA

ID NAME FILT /

In response to the routine name, there were three extra possible responses, and one of the above was slightly altered:

/ - delete the whole routine;

N - skip this routine, go on to the next one;

BS - return to the previous routine;

EN - finish altering this process record.

The output then continued, allowing the output addresses for the routine to be altered.

OUTPUT ADDRESSES

CHANNEL # 0 / <RETURN>

ROUTINE 2 / <RETURN>

PARAMETER 3 / <RETURN>

RECORD ID / <RETURN>

CURRENT OUTPUT 1016.0

PARAMETERS

0.8 / <RETURN>

Displaying the current output, meant the engineer could use his interface to closely examine the processing within control loop and diagnose a problem, if one existed.

As each record was created or altered, it was stored on the backup disc and transmitted to either a control station or the operator interface. Because it was originally intended to have the operator's and engineer's interfaces running concurrently on the PDP, the available colour graphics terminal was not used for the engineer's interface. The sequential input style that resulted from use of the VT100, while proving adequate, meant that the engineer had no overview of the control loop, in particular, that he was configuring or examining. This required that all input be first written down, or the configuring files printed later, to reduce load on his short term memory. Within this restriction, imposed by the terminal, the operation of the interface was satisfactory.

The large number of process routines offered, and the very flexible output addressing scheme for the result of a routine, enabled a wide range of control schemes to be modelled. As discussed in Chapter 4, however, the interface flexibility resulted in a system that was difficult to learn. Engineers needed a long training time (several hours) to become familiar with the method of data input for a control loop, and several months of operation to take full advantage of the flexibility offered by the process routines and addressing scheme. In view of the long learning time, it was felt that training in a problem oriented programming language (e.g. RTL (4-4)) and allowing the engineer to program his control scheme, would have actually increased the flexibility without paying any penalty in interface simplicity.

8.5 Operator's Interface

Hardware:

The operator's interface was also based in the department's PDP11 minicomputer. The hard disc was used for storage of data files for the loop trend displays. The display medium for the interface was an ISC 8001

colour graphics terminal. It had the capability of displaying eight colours. The graphics resolution of the terminal was 192 points vertically, by 160 horizontally. For alphanumeric data it could display 80 characters by 48 lines (24 lines for double height characters). For graphics each character space was divided into four rows of two dots. The colour of each dot within a character space could not be set separately, the status (including blinking, foreground and background colour) was set for all eight dots. This restriction, the terminal's low plotting speed (for display updates) and the low resolution proved to be problems for the interface. Input was by either an infrared light beam array on the terminal screen, to implement a touch sensitive CRT, or a keyboard. The keyboard had eleven backlighted momentary contact push buttons and three indicators. The light beam array was 35 vertically by 47 horizontally, with the grid spacing set at 6 mm. A signal was sent to the computer only if both an x and a y beam were blocked. The touch pad design is shown in Appendix A and the keyboard in Appendix B.

Software:

The operation of the interface is shown in the state diagram (Fig. 8.14). The permitted actions were set by the current display. There were four possible displays:

i) the plant overview display

ii) the subgroup display

iii) the loop display

iv) the hardware status display.

Each display had a top line that contained the blinking cursor, a title, the date, and the time. The cursor was visible to indicate to the operator that graphics terminal was operative. The time was updated every second by the computer and this showed the operator that the computer and the computer-terminal communication line were active and operational. Both

these facts increased operator confidence in his interface.

The overview display (Fig. 8.9) was titled 'Plant Overview Display'. It contained up to eight subgroup symbols, in two rows of four, that each consisted of up to eight control loops. That meant that the operator was able to oversee 64 loops. The low resolution of terminal in fact meant SEE ERRATA that even with only 64 loops, the changes in each loop needed to significant (of the order of 10%) before the shape of the process variable polygon altered noticeably. The value of the process variable was indicated by the distance out from the centre of the symbol of the green polygon on each spoke. The eight spokes were spaced at 45^{o} around the symbol. The two red circles showed the upper and lower absolute alarm limits. If the process variables in a subgroup were all at their respective set points, the green polygon formed a circle between the two red circles. To achieve this, the scale factor was different depending on whether the process variable was greater or less than the set point. The subgroup symbols were headed by the subgroup identification codes. If one of the variables in a subgroup has exceeded an absolute constraint the identification code would blink in red. If the alarm was only a deviation alarm, the code would blink in white. To examine a subgroup in more detail, the operator pointed at the symbol and the subgroup display would appear. The hardware status square at the bottom left enabled the operator to call up the hardware status display.

The title for the subgroup display was taken from the subgroup description. Here, up to eight loops were each represented by a symbol similar to a conventional analog controller display (Fig. 8.10). At the top and bottom of the symbol the two red areas represented the high and low absolute alarm limits respectively. If the process variable had exceeded one of these limits, the red area would flash. The white T symbol showed the set point and deviational alarm limits. The vertical bar showed the alarm limits and the horizontal bar, the setpoint. The

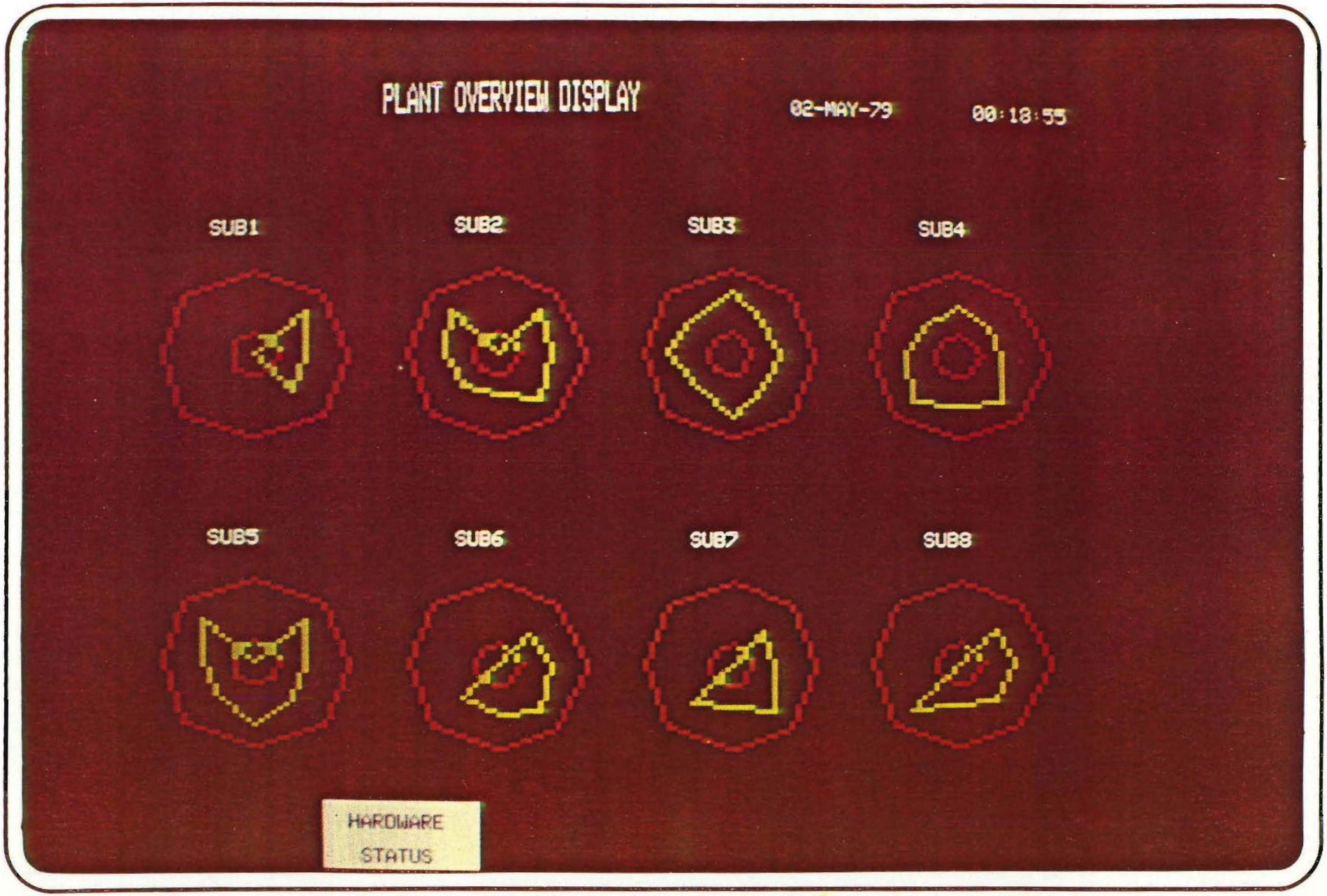


FIG 8.9 OVERVIEW DISPLAY

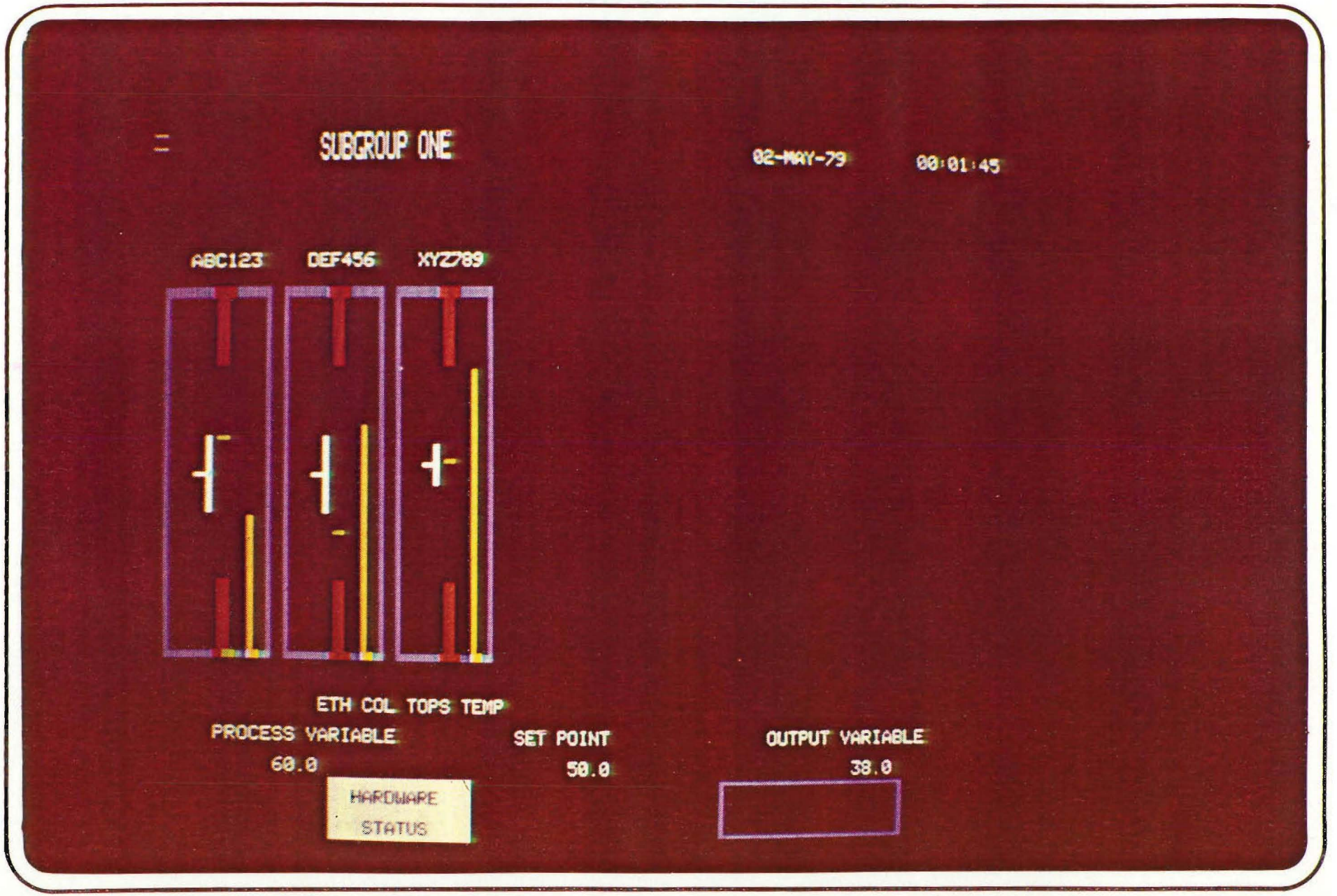


FIG 8.10 SUBGROUP DISPLAY

process variable was indicated by the green bar. To the right, the yellow line showed the position of the output variable, provided output limits had been set to allow scaling. At the top of each symbol was the loop identification code. To select a loop, the operator pointed at that loop's symbol. In the space below the loop symbols, this resulted in the loop description and the digital values of the process variable, the set point and the output variable being written. The light on the ENTER button (see the keyboard display, Fig. 8.13) being turned on. If the enter button was then pressed, the loop display would be called up. While the subgroup display was current, the set point or the output variable (if the controller was in manual) could be changed. To do this, the operator pointed at the variable he wished to alter. The value would be written into the workspace, the blue box on the bottom right of the display, and in the display the value would be written in reverse video (black on white instead of white on black). The CLEAR ENTRY button light would also be activated, to allow the operator to end the sequence before the variable had been altered. To alter the value, the five white buttons were used. They allowed the operator to increase or decrease the variable slowly or rapidly (.1% or 10% steps), or directly input zero. On pressing ENTER, the display would be updated, the change would be logged on a printer and then the new value would be sent to the control station. The white square on the bottom left again allowed access to the hardware status display.

The loop display (Fig. 8.11) used the loop description as a title. It had five areas:

i) the trend display;

ii) a status line giving alarm information;

iii) a digital display of loop parameters;

iv) loop status data;

v) the workspace and hardware status display square.

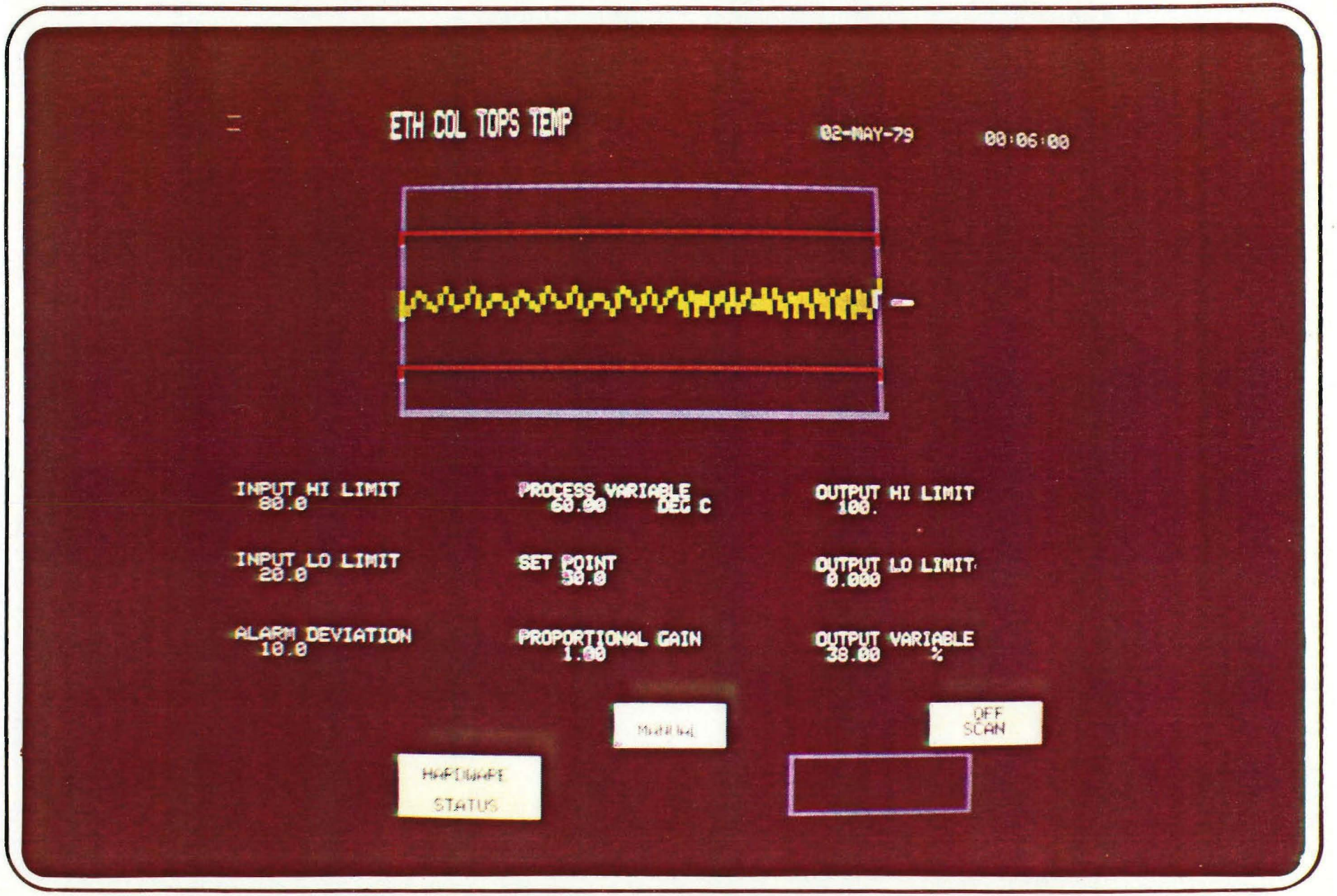


FIG 8.11 LOOP DISPLAY

The trend display showed the previous three and a half hours of data. The red lines at the top and bottom showed the value of the high and low absolute alarms and the white marker to the right, the current value of the set point. Again, because of the resolution of the screen, the trend display tended to be too coarse to be useful. Each vertical point represented 2% of full scale and one horizontal point, approximately 2.5 minutes. The horizontal scale could have been expanded, but there was not enough room in the display to expand the vertical scale.

The alarm status line gave only simple data, stating whether an alarm was an absolute or deviation limit violation. In a system that included a priority alarm analysis, however, it could include data such as time of alarm and whether or not it was a primary or secondary alarm.

The variables for which the digital values were displayed were the high and low absolute and deviation alarm limits. In the next column was the process variable, the set point, and the controller gain. The third column showed the output limits and the output variable. The variables that could be altered, following the sequence described for the subgroup display, were, the deviation alarm limit, the set point, the controller gain, and the output variable (if the controller was in manual).

There were three status squares (only two if the controller was not cascaded). The first indicated whether the loop was off or on scan, which meant that the input data was being processed by the control station. If it was 'off scan', this was written inside a white square. To change the status to 'on scan', the operator pointed at the square. The square would then change to green, with the words 'on scan' written inside it. The controller status square was similar, with white for 'manual' and green for 'auto'. The third square was either 'off or 'on' cascade.

While in the loop display it was also possible to call up other loops, sequentially within the same subgroup, or across subgroups in the same relative position within each subgroup. This was done by using the green 'sequential subgroup' or 'sequential loop' addressing buttons. On pushing one of these buttons, a record identification code would appear in the workspace. While the button was held down, this would slowly cycle through the respective loop codes until the button was released. The enter and clear entry buttons would then light up. Clear entry would just clear the workspace, but enter button would cause a loop display for the new loop to be drawn.

The hardware status display gave a list for each item of hardware in the system; the main computer (or engineer's interface), the operator's interface, and the control station's (one line for each). This list gave their highway addresses and their status; OK, hardware fault or highway fault (see Fig. 8.12). If the status was OK, the line would be in white; if the status was not OK, the line would be in red and the unit name would blink. To notify the operator when a status change occurred, the hardware fault indicator would light up. The fault would also be logged on the printer. To return from the hardware status display, the operator had to press CLEAR DISPLAY button. This button was always active and at any stage the overview could be called up to diagnose an alarm cause.

Alarm notification involved the display changes already mentioned and in addition the ALARM ACKNOWLEDGE button lights up. If it was an absolute limit violation an audible alarm would sound and pressing the alarm acknowledge button would stop the noise. All changes of alarm status, loops going into alarm or returning to normal, were logged on the printer.

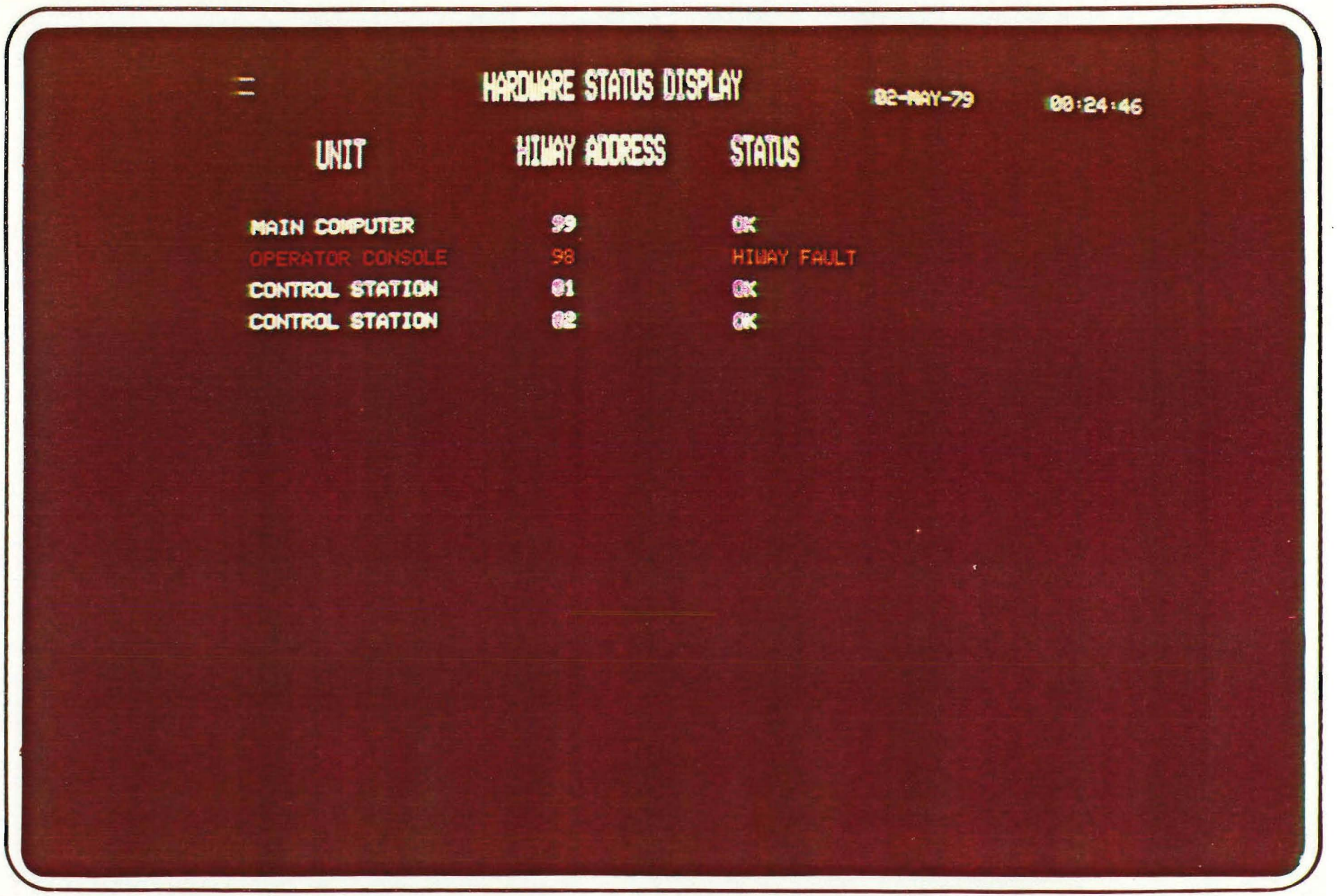


FIG 8.12 HARDWARE STATUS DISPLAY

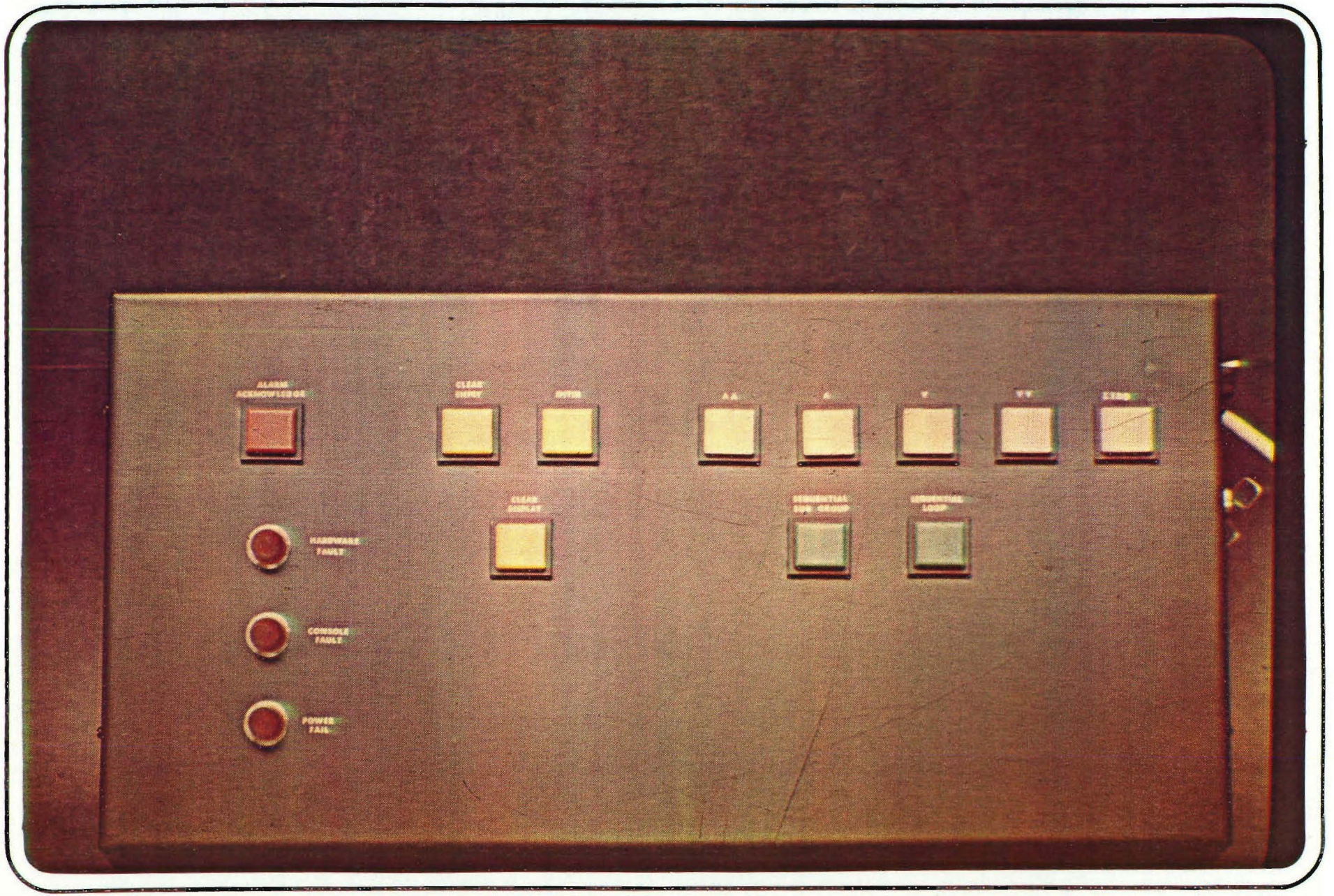
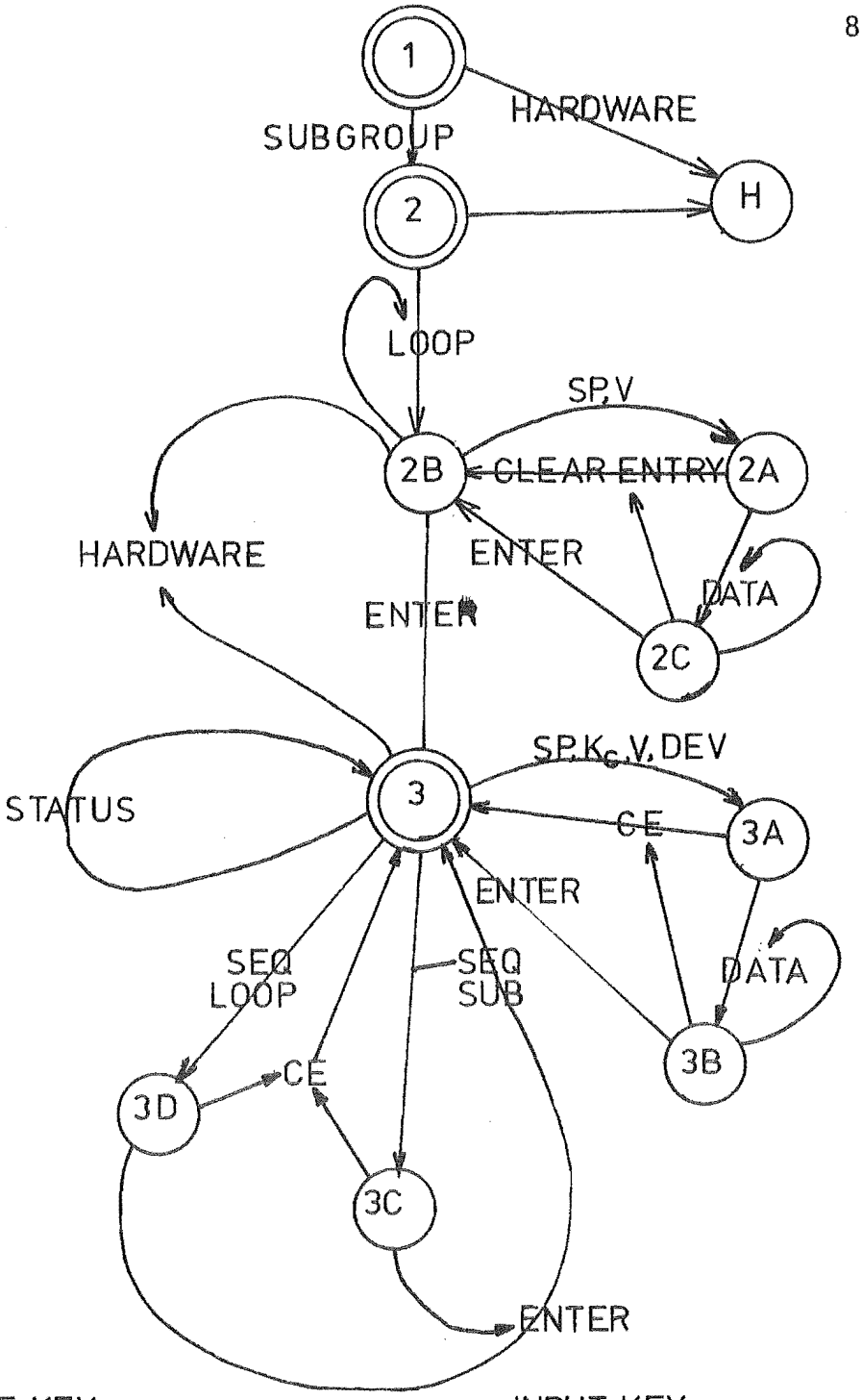


FIG 8.13 KEYBOARD



STATE KEY

1 OVERVIEW DISPLAY
2 SUBGROUP ,,
3 LOOP ,,
4 HARDWARE ,,

OTHERS INTERMEDIATE STATES

INPUT KEY

SP SETPOINT
V OUTPUT VARIABLE
K_c PROCESS GAIN
DEV DEVIATION LIMIT
CE CLEAR ENTRY
SEQ SEQUENTIAL ADDRESSING by
SUB SUBGROUP or LOOP

FIG 8.14 INTERFACE STATE DIAGRAM

All logged data showed the date and time. In addition it showed:

i) for a change to a variable -
record ID, routine, parameter, old value, new value;

ii) for a hardware status change -
unit, highway address, status, destination address;

iii) for an alarm -
record, routine, alarm type.

The destination address was included in the hardware status message to help diagnose a highway fault. The routine was included in the alarm log because a computation error in the process interface also resulted in an alarm message being sent. This would occur if a parameter in a routine was outside its permitted range (see the discussion on the engineer's interface).

The operator interface proved to be very effective. Even with the resolution problems of the graphics terminal, the information presentation proved to be clear and precise. During the design phase, the confidence indicators in the top line of the display proved a boon to debugging. Although this is not the same as operating, it does show they are very quickly assimilated into the sphere of perception. Because only a limited number of active options were open at one time and all others were ignored, and only relevant information was included in each display, training was simplified. An inexperienced computer user (less than one hour's experience) mastered the terminal in under ten minutes of use.

REFERENCES

8-1 Distillation Column Dynamics and Control

G. Wilson, Ph.D. Thesis, University of Canterbury, 1980.

CHAPTER 9

CONCLUSIONS

9.1 Present System

9.2 Future Development

9.1 Present System

A system was designed to meet the specifications set out in Chapters 2-7. This resulted in an effective system, except where economics constrained the hardware implementation.

The communication system was reliable, transmitting messages from the short 10 byte single data value to the three part, 250 bytes per section, control loop record. It would have benefitted from a higher line speed and a hardware implemented bit oriented protocol, but with only one control station this was not a major problem.

The process interface was designed using a single CPU for both control and communication. This caused an occasional clash between the two. The communication system would be in command when the control interrupt occurred. The resultant holdup was slight, however, and the waiting control interrupt would be serviced before the next one arrived.

Because of the large numbers of functions and the output addressing scheme, the engineer's interface was extremely flexible. The ability to format process logs and reconfigure the plant overview display, simply, was another excellent feature. The main drawback of the interface was the experience needed to become competent at implementing control schemes. Allowing the engineer to write his own programs would have been more flexible and been only slightly more difficult, if at all. An improvement to the present method would have been the use of a graphical input method (see Fig. 4.1).

The operator's interface fulfilled expectations in all respects. The subgroup symbol, in the plant overview display, while suffering from the resolution problems of the terminal, made effective use of the operator's pattern recognition ability. The loop symbol, in the subgroup display, was excellent. It gave the operator a large amount of information about the

loop, without overwhelming him with detail. The hierarchical display method kept the operator from being burdened by data he did not need, only providing the detail as it was requested.

The input systems made interface operation simple. Using the touch pad for most input, except those that needed to be location coded or were likely to be a hindrance on the screen (e.g. data manipulation), kept the keyboard simple. Because of this simplicity and the error handling procedures, training times were minimal (less than ten minutes for an inexperienced computer user). A useful improvement may have been the addition of a facility to alter the time scale of the trend display. Options would be; one day, one shift, one hour, and possibly ten minutes. This would have made it useful for a greater range of tasks.

9.2 Future Development

Because of rapid changes in computer technology and its associated fields, it is difficult to make even medium term suggestions without their validity being affected by available hardware. However in the short term, improvements could be made in the human interfaces and the process interface.

Process Interface:

With the advent of a suitable and, therefore, generally accepted process control language, the control stations will have the control schemes written as a standard program. This will enable the use of optimal control theory, multivariable rather than multiloop control. It will also enable better process parameter estimation and, thereby, tighter control.

On the hardware side, cheaper and more powerful microcomputers will eventually result in each transducer having its own associated microcomputer, the output of which can be sent to a local multivariable controller, or, if a single loop PID controller is being used, directly to the actuating

element. The connection of large numbers of single loop controllers to the one communication line, will require improvements in this subsystem also. These include hardware implementation of all levels of the protocol and the use of high speed optic fibres. Redundant hardware and throwaway components will increase the MTBF (mean time between failures) to the order of the life of the process plant.

Engineer's Interface:

The primary improvement to the engineer's interface is the development of a suitable process control computer language. With greater familiarity of computer programming among graduating engineers, the block approach to control system design will become less favoured. Voice input and output, along with improved graphics facilities, will also speed up configuration and initialisation.

Operator's Interface:

In the display area of the operator's interface, an improved overview display would provide the greatest benefit to current systems. The loop and subgroup displays are adequate in terms of operator experience. Voice input or output will only be useful in quiet control rooms. In the process area, the sound would be masked by other voices and general process noise. The use of optimal multivariable controllers will directly affect the operator. Experience will need to be gained so that he does not lose touch with what is occurring in the process. However, the self-tuning capability will shift his task from one of overlooking both the control system and the process, to concentrating entirely on the process.

APPENDIX A

TOUCH PAD DESIGN

Figures

The touch pad was based on a grid of infrared light beams. The light beams, generated by Philips CQY58 LED/BPW22 phototransistor pairs, were at 6 mm spacing, giving an array of 39 x 47 for the ISC terminal screen.

The screen was scanned once every half second, by using a 100 Hz clock to turn the beams off and on. To avoid beam divergence problems only one beam, the LED and corresponding phototransistor, were turned on at once. This also allowed all the receivers to be OR-wired into the one amplifier/filter/detector unit. The switching was accomplished by using a 1 of 16 decoder fed from a counter. This counter also provided the output, to the computer, of the beam that had been broken. Ambient light, sunlight or room lighting, was avoided by modulating the LED power supply at 3 kHz and using a high pass filter in the detector. An audible beep sounded whenever a beam in both axes was broken, indicating the presence of some object. A 'ready' signal was used to interrupt the PDP11 and the touch pad system locked until it received an 'acknowledge' in response.

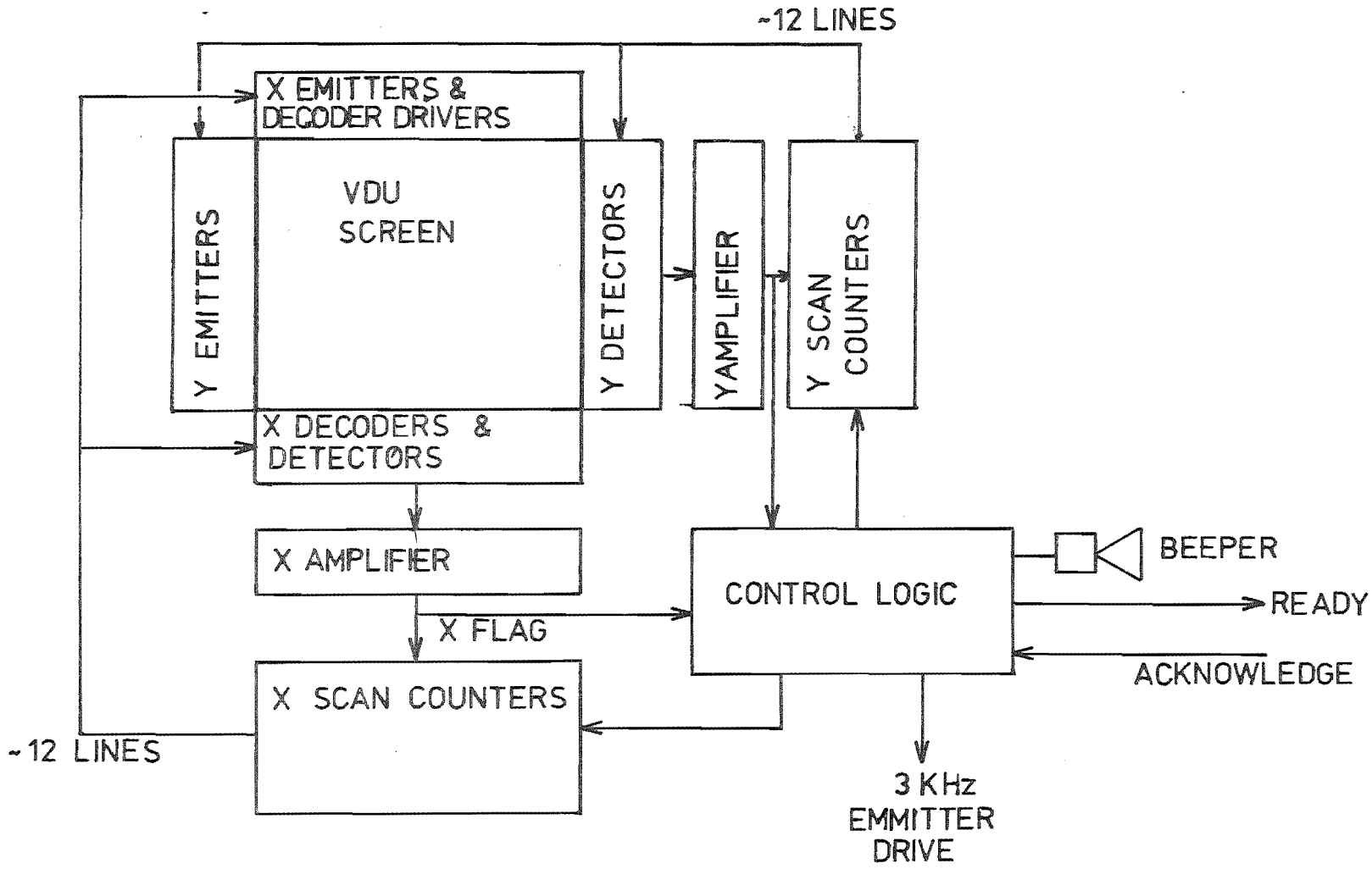


FIG A.1 TOUCH PAD SCHEMATIC DIAGRAM

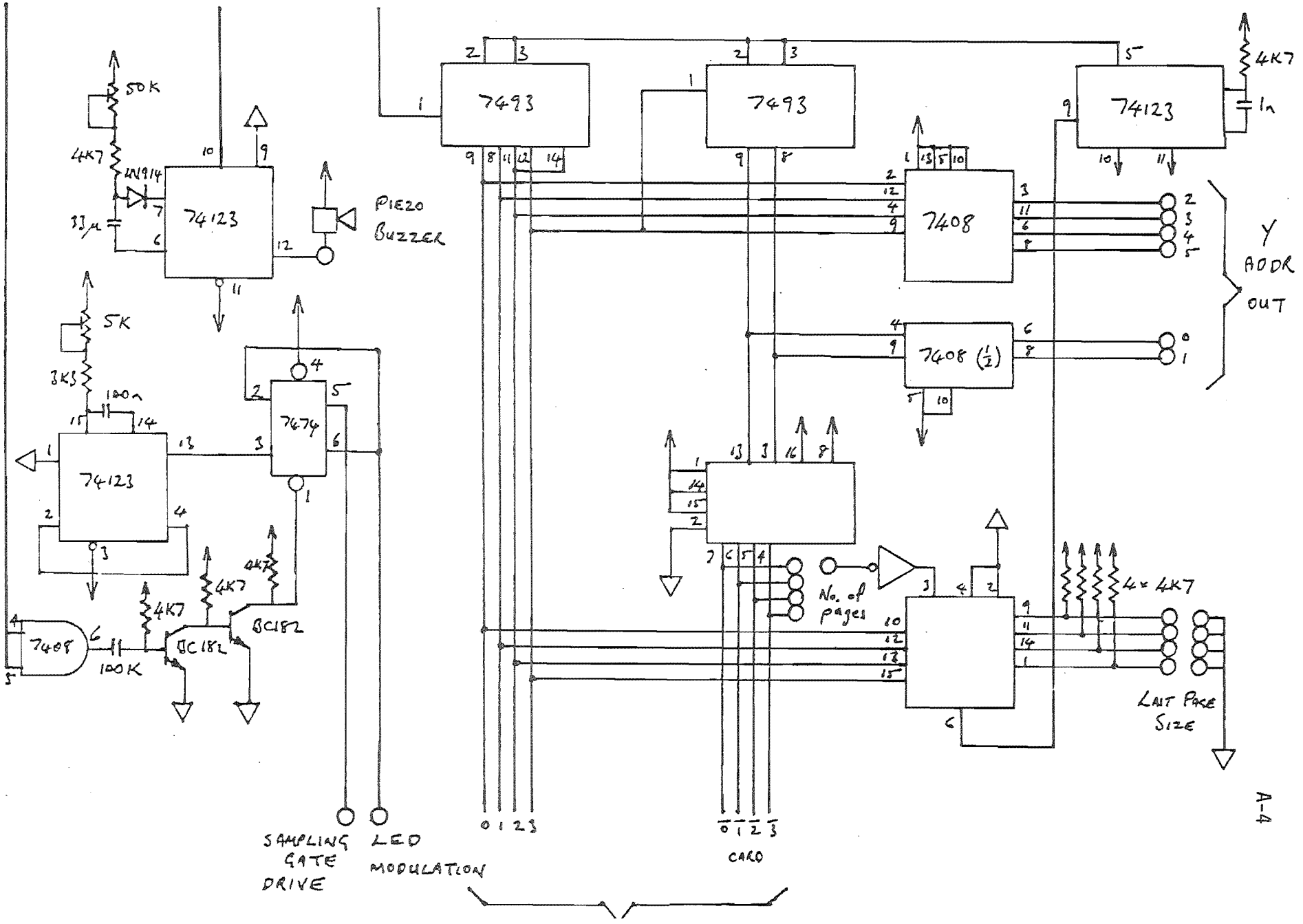


FIG A.2 YAXIS CIRCUIT

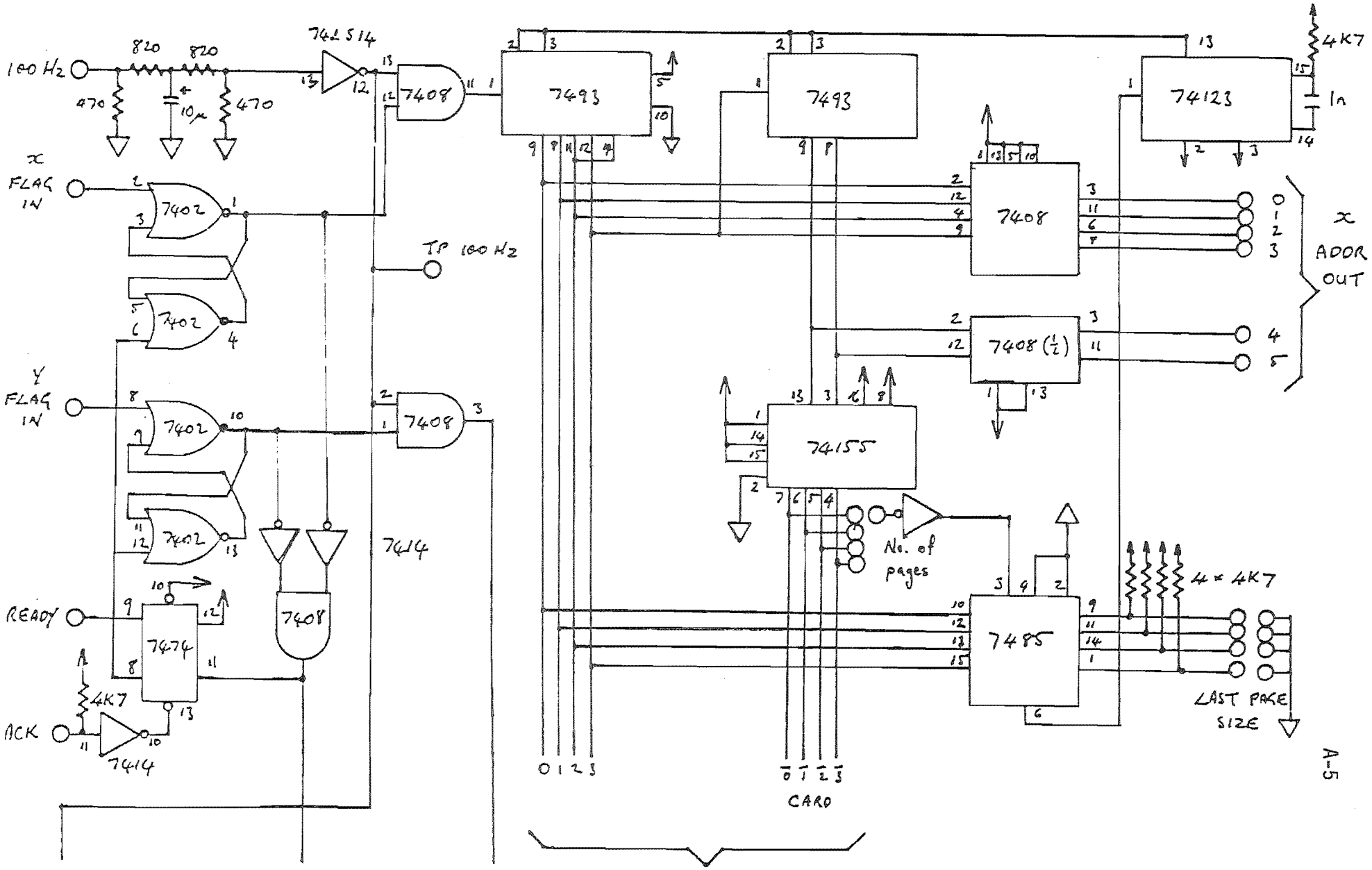


FIG A.3 X AXIS CIRCUIT

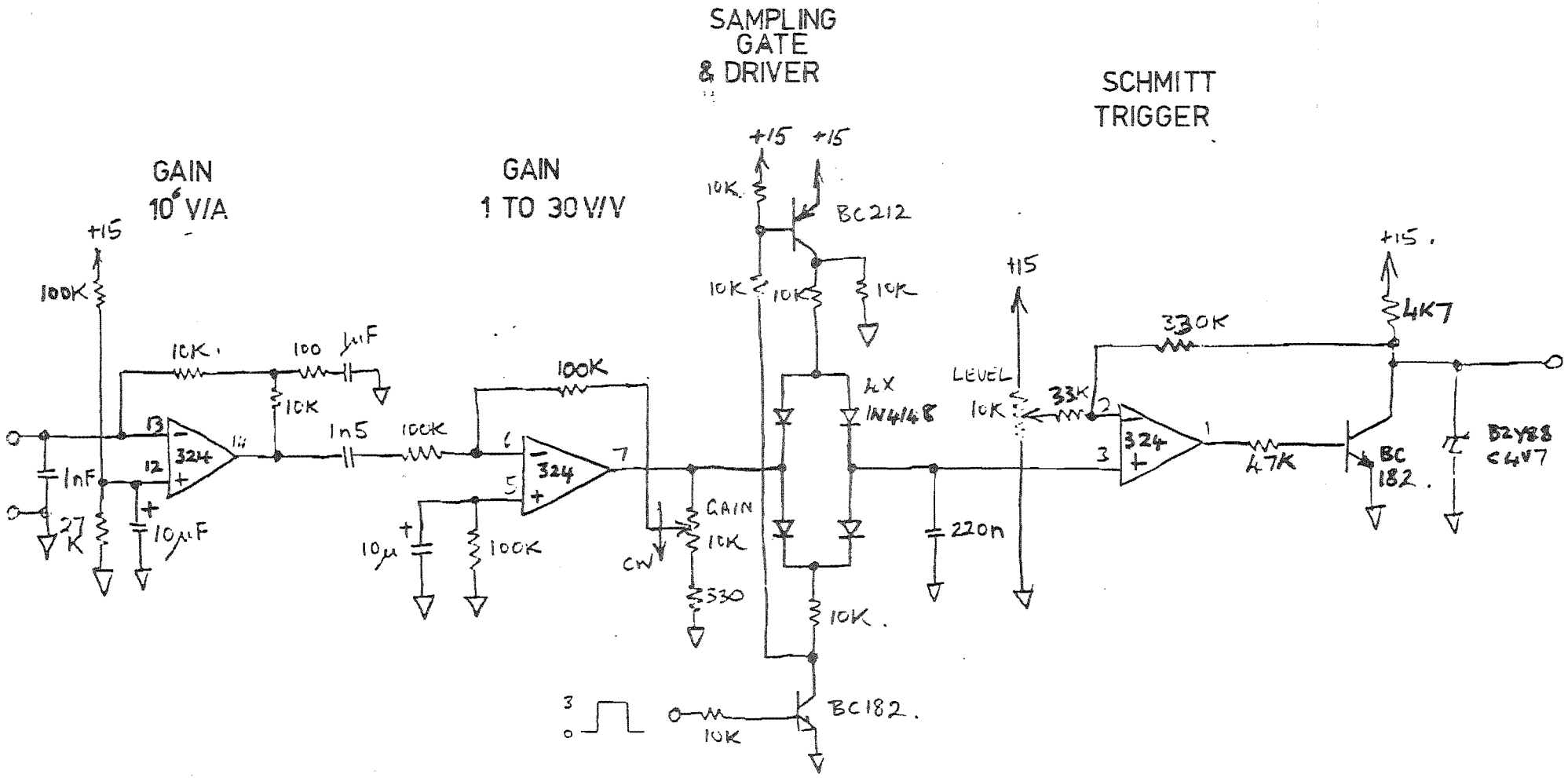


FIG A.4 DETECTOR CIRCUIT

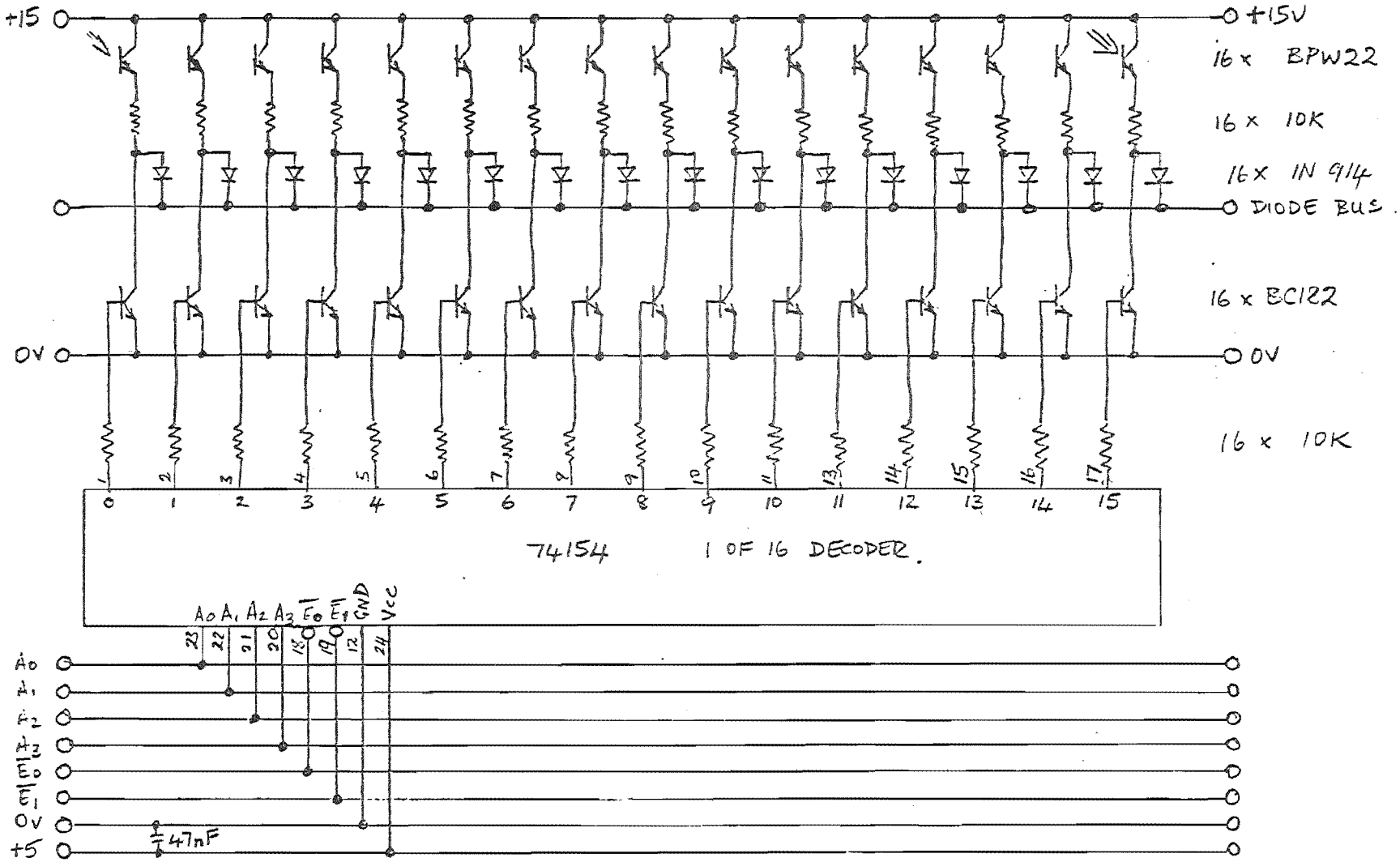


FIG A.5 RECEIVER BOARD CIRCUIT

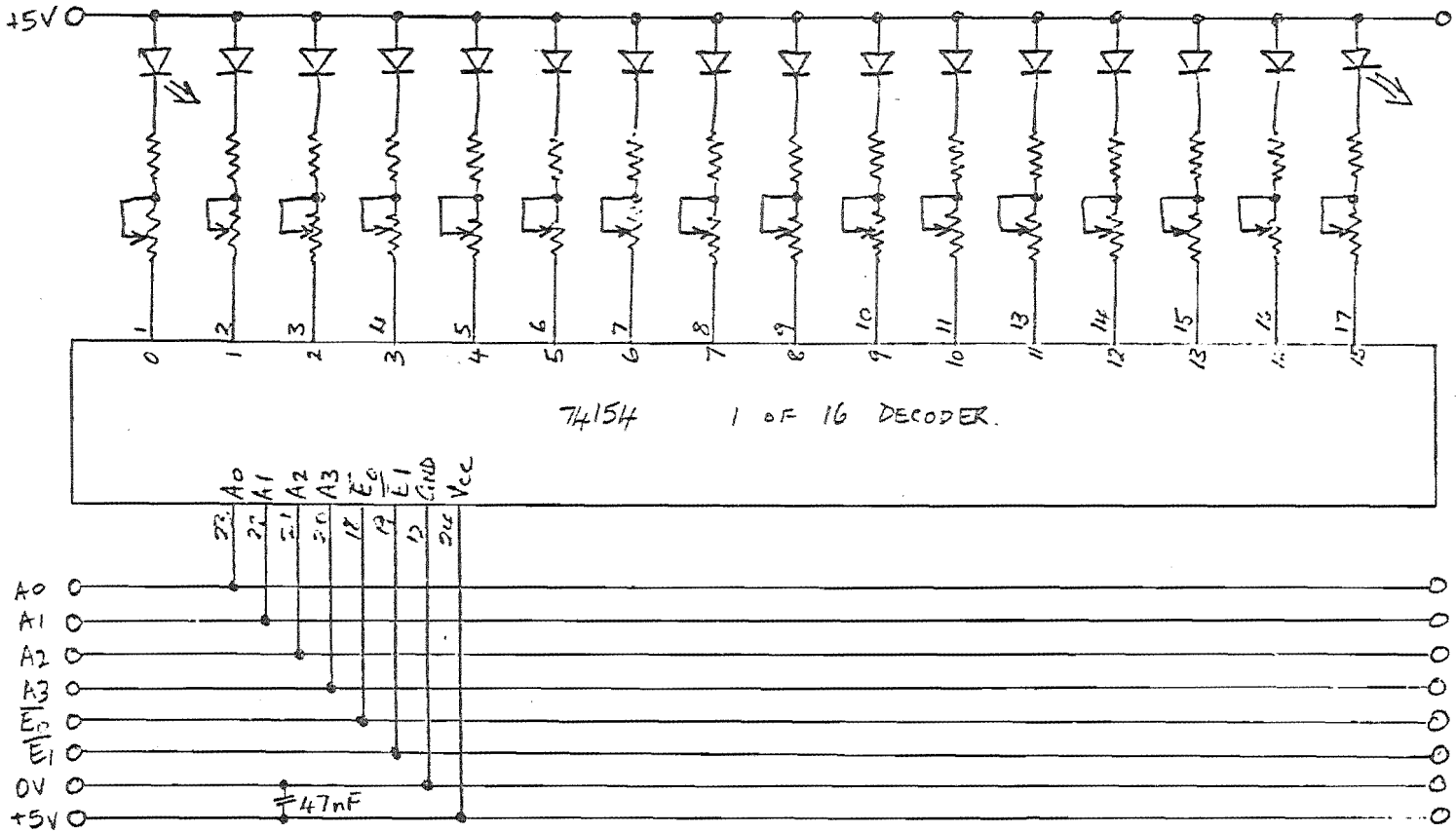


FIG A.6 EMITTER BOARD CIRCUIT

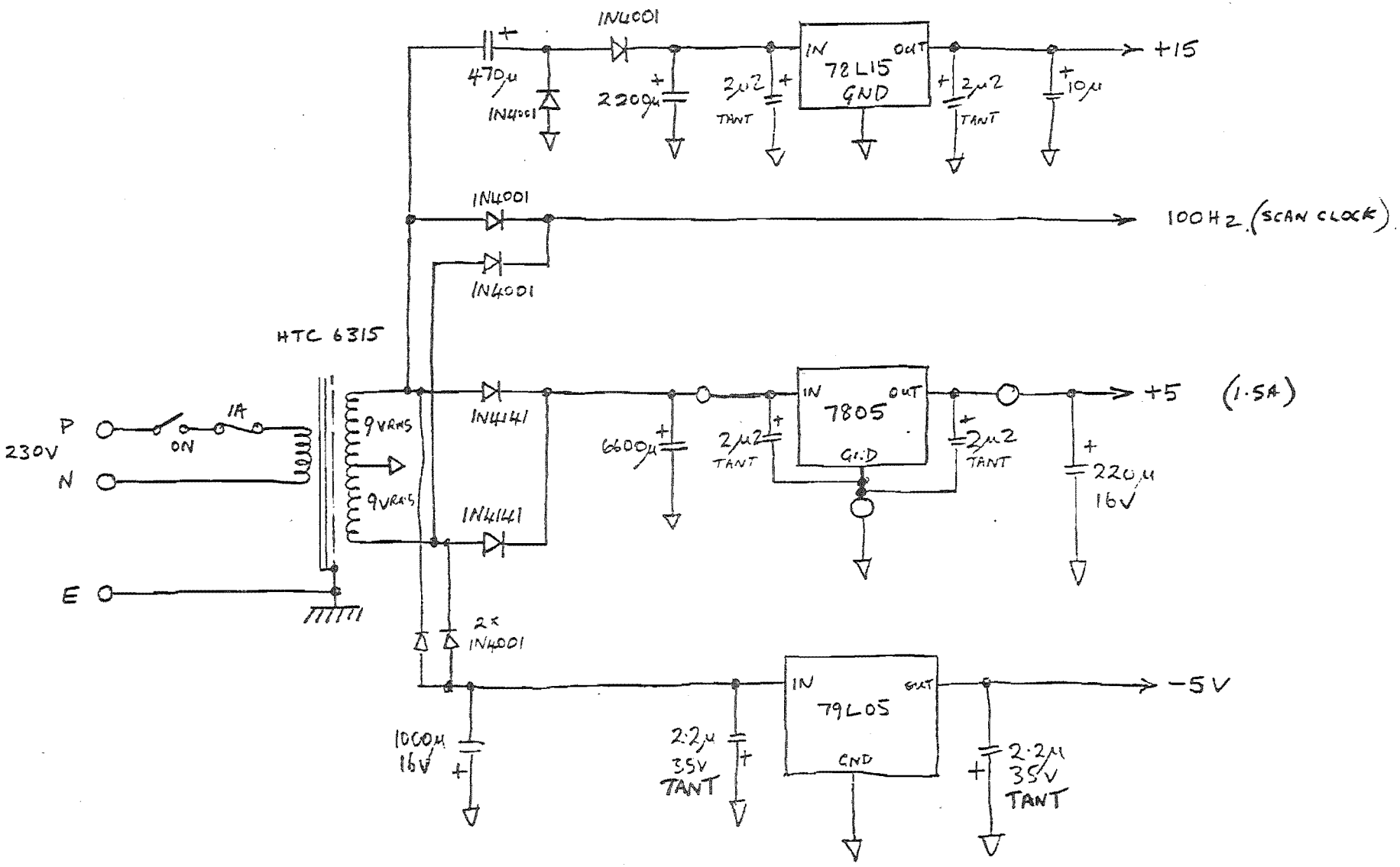


FIG A.7 POWER SUPPLY CIRCUIT

APPENDIX B

KEYBOARD DESIGN

Figures

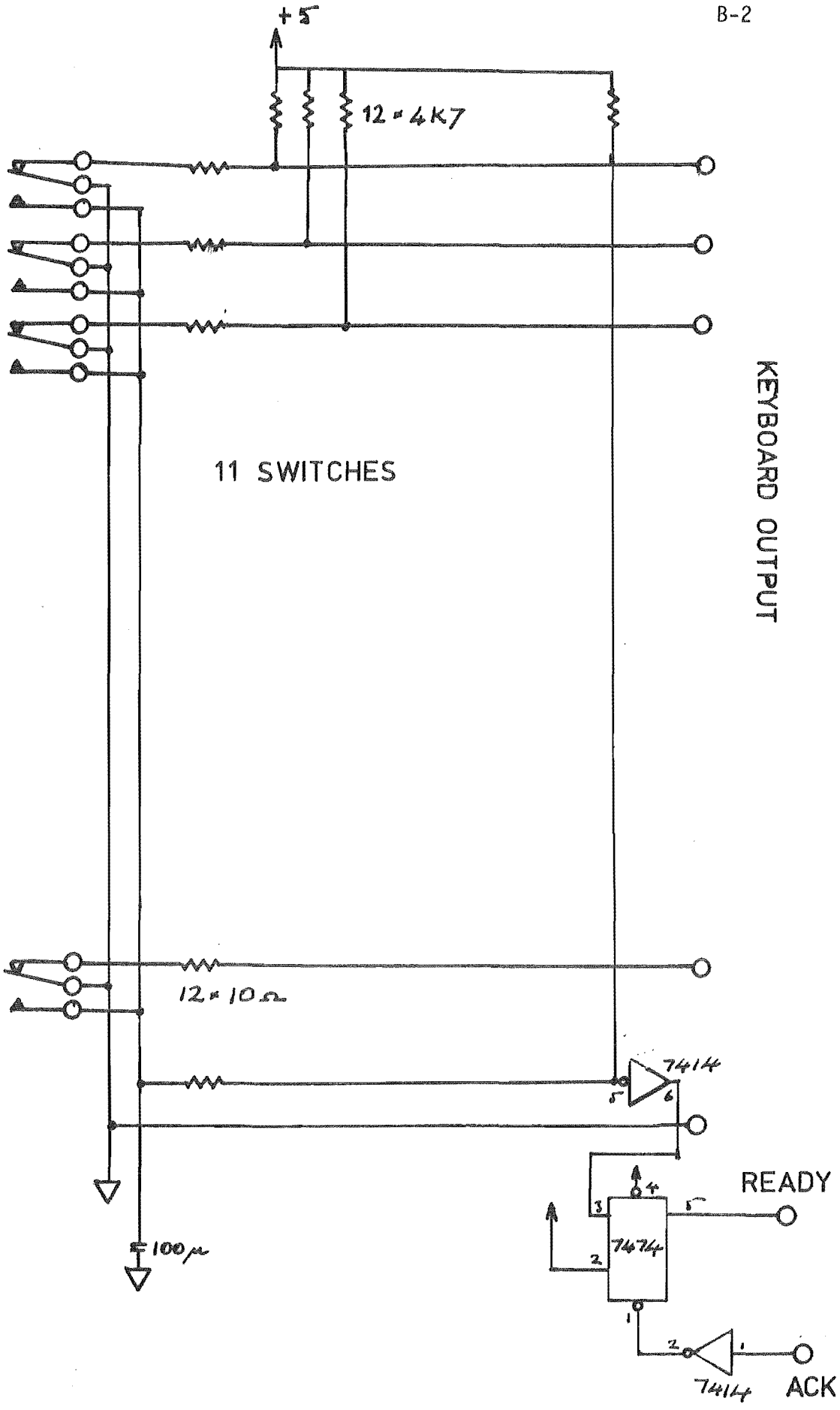
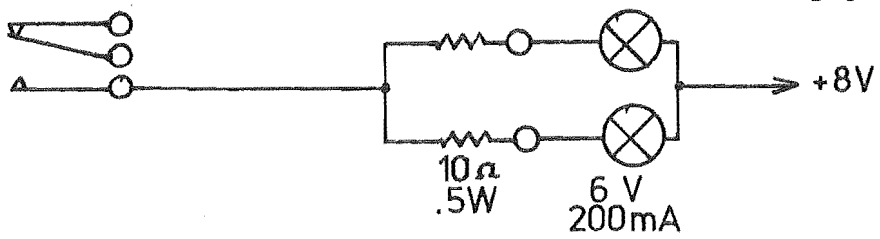


FIG B.1 KEYBOARD SWITCH CIRCUIT



8 OFF

FIG B.2 SWITCH SELF ILLUMINATION WIRING

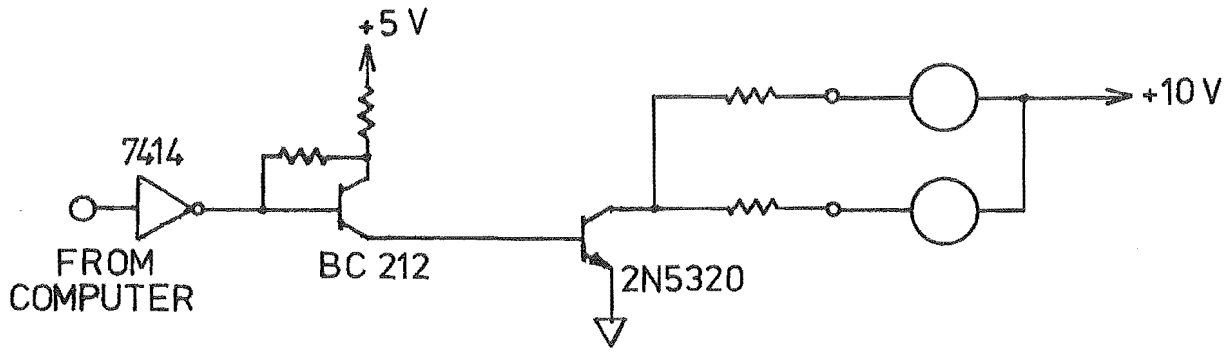


FIG B.3 INDICATOR WIRING

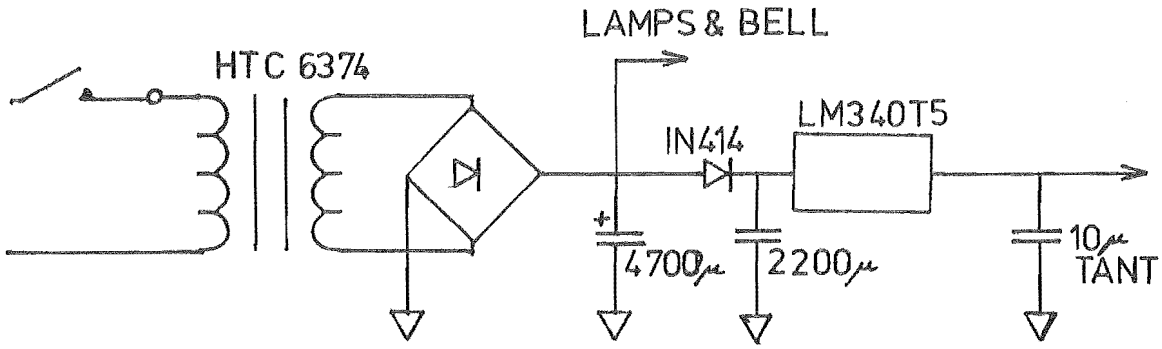


FIG B.4 POWER SUPPLY CIRCUIT